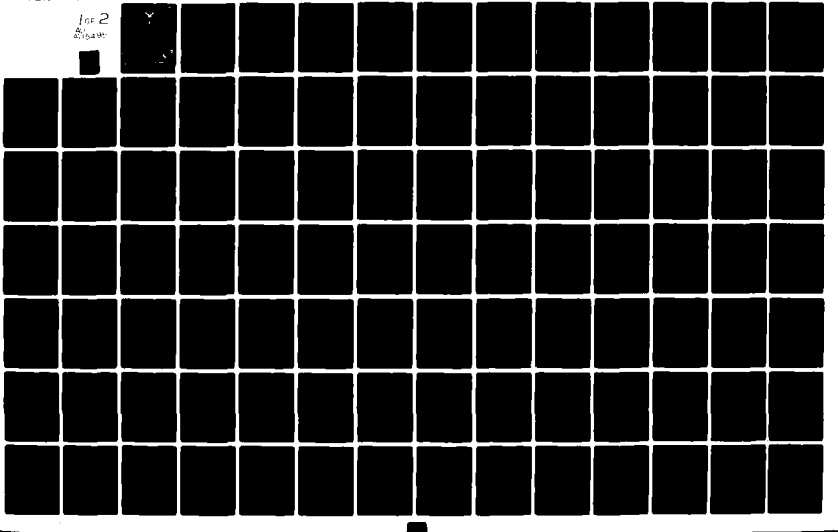
AD-A115 495 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 20/13
CONVERGENCE AND ERROR CRITERIA OF ITERATIVE NUMERICAL SOLUTIONS--ETC(U)
MAR 82 R A WARREN
UNCLASSIFIED AFIT/GNE/PH/82M-12 NL



DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GNE/PH/82M-12

(1)

CONVERGENCE AND ERROR CRITERIA OF ITERATIVE NUMERICAL SOLUTIONS TO THE TRANSIENT HEAT CONDUCTION EQUATION

THESIS

AFIT/GNE/PH/82M-12 Robert A. Warren
Capt USAF

Approved for public release; distribution unlimited.

AFIT/GNE/PH/82M-12

CONVERGENCE AND ERROR CRITERIA OF ITERATIVE NUMERICAL SOLUTIONS TO THE TRANSIENT HEAT CONDUCTION EQUATION

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by
Robert A. Warren, B.S.
Capt. USAF
Graduate Nuclear Engineering
March 1982

Acknowledgments

I would like to express my appreciation to Dr. Bernard Kaplan of the Air Force Institute of Technology for his help and guidance and to Dr. W. Kessler of the Air Force Materials Laboratory for sponsoring this research project. I am indebted to my wife Rita and my daughter Jennifer for being understanding and supportive while I was working on this project. I would also like to thank my wife for the superb typing that transformed a mass of incoherent notes into a readable document.

Robert A. Warren

Contents

List of Figures

List of Tables

Abstract

Full implicit finite differencing was used to approximate transient heat conduction in two spacial dimensions. The resulting set of linear simultaneous equations was solved using successive over-relaxation iterative methods. Errors between the exact matric solution and the iterated solution were computed and compared with several methods of determining error approximations that use the displacement between iterations and an associated parameter. Fundamental parameters of the iteration matrix, such as the spectral radius, were tested as parameters in the error approximation methods. The error methods were compared with respect to accuracy, ease of computing, and computer resources required.

The nodal density used in the numerical approximation to the physical problem was compared with the spectral radius of the resulting iterative matrix problem. The spectral radius increases as nodal density increases.

I Introduction

Background

In practice, few physical problems involving partial differential equations can be solved analytically. Most initial and boundary conditions encountered in practice lack the required symmetry. Alternatively, numerical approximations can be used to generate matric equations that have solutions approximating the physical solution. By increasing the nodal density used in the numerical approximation, the error between the matric and physical solutions is decreased, but at the expense of increasing the number of matric elements.

Iterative techniques are normally used to solve matric equations because iterative techniques are less susceptable to error instabilities, they will not accumulate truncation errors, and they require fewer computer operations, especially if many matric elements equal zero.

Many mathematicians are troubled by the fact that they do not know when to stop the iterative process. The error between iterated and matric solutions is normally unknown.

Problem

The primary purpose of this thesis is to compare methods of determining error limits and approximations with respect to accuracy, computational ease, and required computer resources.

The secondary purpose is to determine what effect increasing the nodal density has on the iterative convergence rate.

Scope

This thesis is limited to the study of transient heat conduction in two spacial dimensions. Only the full implicit numerical approximation to the physical problem was used. The primary iterative method examined was the

successive over-relaxation method.

Approach and Presentation

Section II discusses in detail the theory that is necessary to attempt a solution to the problem of iterative error analysis. First, the matric approximation to the physical problem is explained, followed by the solution of implicit matric equations by iterative techniques. The spectral radius is covered in preparation for examining the convergence properties of iterative matric equations. The concept of norms is introduced to help quantitatively define iterative error limits.

Section III derives the sample problem investigated and then the computer codes are examined which produce error related data.

Section IV presents the numerical data in graphical form and trends are discussed. Both spectral radius versus nodal density and iterative error approximations are discussed.

Section V draws conclusions and gives recommendations based on the data.

II Theory

Matric Approximation to Physical Problem (5:55-56,59, 106-109;9)

In this section we discuss the construction of matric equations that approximate transient heat conduction. The two dimensional transient heat conduction problem can be formulated into the following parabolic partial differential equation.

$$\partial^2 T/\partial x^2 + \partial^2 T/\partial y^2 = (1/K)\, \partial T/\partial t$$

In many instances, this equation with its associated boundary and initial conditions cannot be solved analytically. One method of simplifying this continuous problem, is to make the equation and the domain of the variables discrete. First, a grid is superimposed over the variable domain. The function, T, is then expanded in a Taylor series about selected adjacent grid points. Linear combinations of these are used to form difference approximations to the differentials (5:59). The following are six standard difference operators, the differentials they approximate, and the associated truncation error (5:55-56).

First forward difference

$$\Delta T(x) = T(x+h) - T(h)$$

$$\partial T/\partial x = \Delta T/h + \mathcal{O}(h)$$

First central difference

$$\delta T(x) = T(x+h) - T(x-h)$$

$$\partial T/\partial x = \delta T/2h + \mathcal{O}(h^2)$$

First backward difference

$$\nabla T(x) = T(x) - T(x-h)$$

$$\partial T/\partial x = \nabla T/h + \mathcal{O}(h)$$

Second forward difference

$$\Delta^2 T(x) = T(x+2h) - 2T(x+h) + T(x)$$

$$\partial^2 T/\partial x^2 = \Delta^2 T/h^2 + \mathcal{O}(h)$$

Second central difference

$$\delta^2 T(x) = T(x+h) - 2T(x) + T(x-h)$$

$$\partial^2 T/\partial x^2 = \delta^2 T/h^2 + \mathcal{O}(h^2)$$

Second backward difference

$$\nabla^2 T(x) = T(x) - 2T(x-h) + T(x-2h)$$

$$\partial^2 T/\partial x^2 = \nabla^2 T/h^2 + \mathcal{O}(h)$$

where h is the grid interval in the x coordinate.

Replacing differentials with finite differences, a set of linear difference equations is created. One equation exists for each spacial grid point, or node, interior to the boundary. These equations are then written compactly as a matric equation.

$$u = Av + b$$

where

A = matrix of coefficients

u = vector of known (or unknown) nodal values of the function T

v = vector of unknown (or known) nodal values of the function T

b = vector of known constant elements derived from boundary conditions.

This matric equation is solved once for each discrete time step.

Selection of a difference operator to approximate a given differential is governed by truncation error, error stability and computational ease of the resulting matrix equation. Several combinations of difference approximations exist that transform the parabolic partial differential equation into two basic forms of matric equation. One is the explicit form and the other is the full implicit form.

Each difference equation of the explicit form expresses a single unknown nodal quantity, T, in terms of several known nodal quantities. In the matric equation

$$u = Av + b$$

u becomes the unknown vector and v becomes the known vector. Solution requires simple matric multiplication. Error stability is conditional, depending on the time and space grid intervals.

Each difference equation of the full implicit form expresses several unknown nodal quantities in terms of a single known nodal quantity. In the matric equation

$$u = Av + b$$

u now becomes the known vector and v becomes the unknown

vector. Solution requires a method equivalent to matric inversion. Error stability is unconditional.

A weighted combination of the implicit and explicit forms (5:108-109) can be used to reduce truncation error. The spacial difference operators of the implicit form are weighted by a factor α, while the spacial difference operators of the explicit form are weighted by a factor $1-\alpha$. The combination form is conditionally stable if

$$0 \leq \alpha < 1/2$$

and unconditionally stable if

$$1/2 \leq \alpha \leq 1$$

When α is set equal to ½, the combination form is called the Crank-Nicholson method. Crandall found the α that minimizes the truncation error. Truncation error can also be reduced by decreasing the spacial grid interval.

Solution of the Implicit Matrix Equation

Several methods exist for solving the implicit matric equation

$$A x = b$$

directly. The matrix can be inverted (4:125-129) to form the explicit matrix equation

$$x = A^{-1} b$$

the vector, x, can be found by using Gauss reduction (5:9-11), or if the matrix is block tridiagonal, the Thomas method of matrix factorization (5:46-49) can be used to find x. The Thomas method is a special case of Gauss reduction.

Unfortunately, these direct methods suffer from a

serious flaw. The number of algebraic operations required is proportional to a power of the number of elements in the unknown vector, x. Each algebraic operation produces truncation error which accumulates as the number of operations increase. When the number of unknown vector elements is large, the accumulated error can be significant (5:111).

Alternately, the implicit matric equation

$$A x = b$$

can be rewritten as (12:236)

$$x = G x + c$$

An iterated solution can then be attempted

$$x^{(n+1)} = G\, x^{(n)} + c$$

where

$x^{(n)}$ = nth iterated vector

G = iteration matrix

c = known constant vector

The advantage here is that truncation error accummulated during one iteration, is not passed on to the next. Also, by using an equivalent algorithm, the number of algebraic operations are reduced if some elements of matrix A equal zero (5:111).

We now define the displacement vector as the difference between two successive iterated vectors.

$$d^{(n)} = x^{(n+1)} - x^{(n)}$$

The displacement will be used here to describe the iterative algorithms.

The three iterative methods most commonly used are the Jacobi, Gauss-Seidel, and successive over-relaxation (SOR) methods. Their algorithms are derived by rewritting the implicit matrix in terms of its strictly upper triangular, diagonal, and strictly lower triangular matrices. They are denoted by U, D, and L respectively (12:234).

$$A x = b$$

$$A = U + D + L$$

$$(U + D + L) x = b$$

$$Ux + Dx + Lx = b$$

$$Dx = -Ux - Lx + b$$

The Jacobi method expresses the (n+1)th iterative vector elements exclusively in terms of the nth iterative vector elements (12:228).

$$Dx^{(n+1)} = -Ux^{(n)} - Lx^{(n)} + b$$

$$Dx^{(n+1)} - Dx^{(n)} = -Ux^{(n)} - Dx^{(n)} - Lx^{(n)} + b$$

$$x^{(n+1)} - x^{(n)} = D^{-1}(-Ux^{(n)} - Dx^{(n)} - Lx^{(n)} + b)$$

$$d^{(n)} = D^{-1}(-Ux^{(n)} - Dx^{(n)} - Lx^{(n)} + b)$$

The Gauss-Seidel method attempts to accelerate convergence by using the (n+1)th iterative vector elements as soon as

they become available (10:229).

$$D x^{(n+1)} = -U x^{(n)} - L x^{(n+1)} + b$$

$$D x^{(n+1)} - D x^{(n)} = -U x^{(n)} - D x^{(n)} - L x^{(n+1)} + b$$

$$x^{(n+1)} - x^{(n)} = D^{-1}(-U x^{(n)} - D x^{(n)} - L x^{(n+1)} + b)$$

$$d^{(n)} = D^{-1}(-U x^{(n)} - D x^{(n)} - L x^{(n+1)} + b)$$

The SOR method attempts to further accelerate convergence by multiplying the Gauss-Seidel displacement by an acceleration constant, w (10:230).

$$d^{(n)} = w D^{-1}(-U x^{(n)} - D x^{(n)} - L x^{(n+1)} + b)$$

The iterative matric equations

$$x^{(n+1)} = G x^{(n)} + c$$

that are equivalent to the above algorithms are

$$D x^{(n+1)} = -U x^{(n)} - L x^{(n)} + b$$

$$D x^{(n+1)} = (-U - L) x^{(n)} + b$$

$$x^{(n+1)} = D^{-1}(-U - L) x^{(n)} + D^{-1} b$$

for the Jacobi method,

$$D x^{(n+1)} = -U x^{(n)} - L x^{(n+1)} + b$$

$$D x^{(n+1)} + L x^{(n+1)} = -U x^{(n)} + b$$

$$(D + L) x^{(n+1)} = -U x^{(n)} + b$$

$$x^{(n+1)} = (D+L)^{-1}(-U)\,x^{(n)} + (D+L)^{-1}\,b$$

for the Gauss-Seidel method, and

$$x^{(n+1)} - x^{(n)} = w\,D^{-1}(-Ux^{(n)} - Dx^{(n)} - L\,x^{(n+1)} + b)$$

$$Dx^{(n+1)} - Dx^{(n)} = -w\,Ux^{(n)} - wDx^{(n)} - wL\,x^{(n+1)} + wb$$

$$Dx^{(n+1)} + wL\,x^{(n+1)} = -w\,Ux^{(n)} + Dx^{(n)} - wDx^{(n)} + wb$$

$$(D+wL)\,x^{(n+1)} = (-wU + D - wD)\,x^{(n)} + wb$$

$$x^{(n+1)} = (D+wL)^{-1}(-wU + D - wD)\,x^{(n)} + (D+wL)^{-1}wb$$

for the SOR method.

Spectral Radius

One must understand the concept of matric spectral radius in order to study the properties of iterative matric solutions. The spectral radius will be defined first. Then a generalized method for finding the approximate spectral radius and a method for determining the accuracy of that approximation will be discussed.

The concept of spectral radius is closely related to the concepts of eigenvalue and eigenvector. If a matrix operates on a vector and results in a scalar constant times the original vector, then that scalar constant is an eigenvalue of the matrix and the vector is an eigenvector associated with that eigenvalue. The equation is an eigenequation.

$$Mx = \lambda x$$

Unless a matrix is square, matrix multiplication will cause the resultant vector to have a dimension different from the original vector. Therefore eigenvalue and eigen-

vectors are associated with square matrices only. The number of eigenvalues associated with a square matrix is equal to the dimension of the matrix. A three by three matrix will have three eigenvalues.

If an eigenvalue differs from all other eigenvalues, it is termed distinct. If an eigenvalue is equal to some other eigenvalue, it is termed multiple. The number of eigenvalues with equal value is termed the eigenvalue multiplicity.

All eigenvectors associated with a given eigenvalue form a vector subspace. The number of linearly independent vectors needed to span that subspace is no larger than the multiplicity of the associated eigenvalue (5:23-31;1:168-171). Associated eigenvectors may not be able to span the vector space for non-symmetric or non-hermitian matrices. Additional principal vectors may have to be added. A distinct eigenvalue has a multiplicity equal to one. All eigenvectors associated with it can be spanned by one eigenvector. The eigenvectors are all scalar multiples of each other. (3:131)

$$Mx = \lambda x$$

$$a\,Mx = a\lambda x$$

$$Max = \lambda ax$$

$$My = \lambda y$$

Two of these eigenvectors have unit length. One is the negative of the other. (9:35)

An eigenvalue having a multiplicity equal to three might need one, two, or three linearly independent vectors to span the eigenvector subspace associated with it. Any linear combination of eigenvectors associated with the same eigenvalue is itself an eigenvector (1:169).

$$a M x = a \lambda x$$

$$b M y = b \lambda y$$

$$a M x + b M y = a \lambda x + b \lambda y$$

$$M a x + M b y = \lambda a x + \lambda b y$$

$$M (a x + b y) = \lambda (a x + b y)$$

If an eigenvalue is not real, then its complex conjugate is one of the other eigenvalues. The associated eigenvector subspaces are complex conjugates of each other (5:23-24).

$$M x = \lambda x$$

$$M x^* = \lambda^* x^*$$

Zero is a perfectly acceptable eigenvalue that is associated with not all matrices. It will have an associated eigenvector subspace distinct from nonzero eigenvalues.

The matric spectral radius can now be defined as the modulus of the maximum modulus eigenvalue,

$$\rho = |\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$$

where

ρ = spectral radius

λ_i = matric eigenvalues

n = dimension of square matrix

Several methods exist that can be used to find the eigenvalues and thus the spectral radius of a given matrix. Each method has advantages and disadvantages. One method is to solve the characteristic equation associated with the matrix.

$$|M - I\lambda| = 0$$

When formulating this equation into standard polynomial form, truncation error associated with calculating the coefficients becomes significant as order is increased (3:130). n factorial terms must be summed to evaluate a determinant of order n (11:622). For example, a determinant of order 20 would require the summation of 2×10^{18} terms! If the resulting equation is quartic, cubic, quadratic, or linear, it may be solved analytically (3:104-107). If it is of higher order, one must resort to iterative techniques.

The eigenvalues of some matrices can be found by inspection. The eigenvalues of diagonal matrices are the diagonal elements. (5:26;1:171-172;3:132)

The power method is used to generate a sequence of iterated vectors (5:140) from which several schemes have been devised to extract the spectral radius and an associated eigenvector. (7:5.1-5.8;5:144;10:460) The power method consists of multiplying an initial trial vector by the matrix. The result is multiplied by the matrix again and so on. Essentially the initial vector is multiplied by an integer power of the matrix.

$$M x_0 = x_1$$

$$M x_1 = x_2$$

$$\vdots$$

$$M x_{n-1} = x_n$$

$$M^n x_0 = x_n$$

The following discussion will be limited to real symmetric matrices although the results can be generalized to Hermitian and other special matric catagories. All eigenvalues associated with real symmetric matrices are real. A set of n orthogonal eigenvectors exists that spans the vector space associated with the n by n matrix. (5:24-25;1:168-169)

The spectral radius of a matrix is estimated by using the generalized quotient method. An initial vector is expanded in terms of the eigenvectors.

$$x^{(0)} = c_1 e_1 + c_2 e_2 + \cdots + c_n e_n$$

where

c_i = component coefficients

e_i = eigenvector

n = matric dimension

$x^{(0)}$ = initial trial vector

The kth iterated vector is formed.

$$x^{(k)} = M^k x^{(0)}$$

$$= M^k \sum_{i=1}^{n} (c_i e_i)$$

$$= \sum_{i=1}^{n} (c_i M^k e_i)$$

$$Me = \lambda e$$

$$x^{(k)} = \sum_{i=1}^{n} (c_i \lambda_i^k e_i)$$

The eigenvalues can be ordered such that

$$|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$$

Assume that the maximum modulus eigenvalue is of multiplicity $s \geq 1$. Only if the negative of the maximum modulus eigenvalue is not an eigenvalue itself

$$-\lambda_1 \neq \lambda_i \qquad \text{for all i}$$

can the spectral radius be approximated by forming a quotient using two vectors that are separated by one iteration.

$$x^{(k+1)} = \lambda_1^{k+1} \left[\sum_{i=1}^{s} (c_i e_i) + \sum_{i=s+1}^{n} (c_i [\lambda_i/\lambda_1]^{k+1} e_i) \right]$$

$$x^{(k)} = \lambda_1^{k} \left[\sum_{i=1}^{s} (c_i e_i) + \sum_{i=s+1}^{n} (c_i [\lambda_i/\lambda_1]^{k} e_i) \right]$$

Since

$$|\lambda_i/\lambda_1| < 1 \qquad \text{for} \qquad i > s$$

then

$$\lim_{k\to\infty} [\lambda_i/\lambda_1]^{k+1} = \lim_{k\to\infty} [\lambda_i/\lambda_1]^{k} = 0$$

and as k increases toward infinity

$$\lambda_1 \simeq v^T x^{(k+1)} / (v^T x^{(k)}) \equiv q_1$$

where v is an arbitrary vector. In the limit

$$\lambda_1 = \lim_{k\to\infty} [\, v^T x^{(k+1)} / (v^T x^{(k)}) \,]$$

Assume that an eigenvector of multiplicity $r \geq 0$ is equal to the negative of the maximum modulus eigenvector.

$$-\lambda_1 = \lambda_i \qquad \text{for} \qquad i = s+1,\ s+r$$

The spectral radius can always be approximated by forming a quotient using two vectors that are separated by two iterations

$$x^{(k+2)} = \lambda_1^{k+2} \Big[\sum_{i=1}^{s} (c_i e_i) + (-1)^{k+2} \sum_{i=s+1}^{s+r} (c_i e_i) + \sum_{i=s+r+1}^{n} (c_i [\lambda_i/\lambda_1]^{k+2} e_i) \Big]$$

$$x^{(k)} = \lambda_1^{k} \Big[\sum_{i=1}^{s} (c_i e_i) + (-1)^{k} \sum_{i=s+1}^{s+r} (c_i e_i) + \sum_{i=s+r+1}^{n} (c_i [\lambda_i/\lambda_1]^{k} e_i) \Big]$$

Again

$$\lim_{k\to\infty} [\lambda_i/\lambda_1]^{k+2} = \lim_{k\to\infty} [\lambda_i/\lambda_1]^{k} = 0$$

and as k increases toward infinity

$$\lambda_1^2 = v^T x^{(k+2)} / (v^T x^{(k)})$$

In the limit as k approaches infinity

$$\lambda_1^2 = \lim_{k\to\infty} [\, v^T x^{(k+2)} / (v^T x^{(k)}) \,]$$

Note that if $r \geq 1$

$$\lambda_1 \neq \lim_{k\to\infty} [\, v^T x^{(k+1)} / (v^T x^{(k)}) \,]$$

because

$$(-1)^{k+2} = (-1)^{k} \neq (-1)^{k+1}$$

(7:5.1-5.8;10:460-462;5:144)

In normal practice, the kth iterated vector is scaled in some way before iterating to find the next iterated vector. In this way working with very small or very large vector elements is avoided. For example, the largest vector element or the vector length can be normalized to unity (5:145;7:5.4).

The choice of the vector v affects the quotient convergence rate. The quickest convergence is realized when v is chosen such that

$$q_1 = [x^{(k+1)}]^T x^{(k+1)} / ([x^{(k+1)}]^T x^{(k)})$$

and

$$q_2 = [x^{(k+2)}]^T x^{(k+2)} / ([x^{(k+2)}]^T x^{(k)})$$

the first being the Rayleigh quotient (5:144;10:460; 7:5.1-5.8).

Kreyszig (10:462-463) gives a method for determining an error limit for the approximation $\lambda_q \simeq q_1$, when $v=x^{(k)}$. Where λ_q is the eigenvalue closest to the quotient approximation. Note that λ_q may not equal λ_1 for early iterations, but will eventually. The following is a generalization of that method applicable to both $\lambda_q \simeq q_1$ and $\lambda_q^2 \simeq q_2$ and for any arbitrary vector v. Derivations are given in Appendix A. For $\lambda_q \simeq q_1$ the error is limited by

$$\epsilon_1 \leq \sqrt{(q_1^2 - 2q_1 m_1/m_0 + m_2/m_0)} \equiv \delta_1$$

where

$$\epsilon_1 \equiv |\lambda_q - q_1|$$

$$q_1 \equiv V^T X^{(k+1)} / (V^T X^{(k)})$$

$$m_0 \equiv [X^{(k)}]^T X^{(k)}$$

$$m_1 \equiv [X^{(k)}]^T X^{(k+1)}$$

$$m_2 \equiv [X^{(k+1)}]^T X^{(k+1)}$$

For $\lambda_q^2 \simeq q_2$ the error is limited by

$$\epsilon_2 \leq \sqrt{(q_2^2 - 2 q_2 m_3/m_0 + m_4/m_0)} \equiv \delta_2$$

where

$$\epsilon_2 \equiv |\lambda_q^2 - q_2|$$

$$q_2 \equiv V^T X^{(k+2)} / (V^T X^{(k)})$$

$$m_0 \equiv [X^{(k)}]^T X^{(k)}$$

$$m_3 \equiv [X^{(k)}]^T X^{(k+2)}$$

$$m_4 \equiv [X^{(k+2)}]^T X^{(k+2)}$$

For $\lambda_q \simeq \sqrt{(q_2)}$, the error is limited by either

$$\epsilon_3 \leq -\sqrt{(q_2)} + \sqrt{(q_2 + \delta_2)}$$

if $\lambda_q^2 \geq q_2$, or

$$\epsilon_3 \leq \sqrt{(q_2)} - \sqrt{(q_2 + \delta_2)}$$

if $\lambda_q^2 \leq q_2$ and $q_2 \geq \delta_2$, or

$$\epsilon_3 \leq \sqrt{(q_2)}$$

if $\lambda_q^2 \leq q_2$ and $q_2 \leq \delta_2$, where

$$\epsilon_3 \equiv |\lambda_q - \sqrt{(q_2)}|$$

Iterative Matric Convergence (10:236-237)

Recall that the implicit matric equation

$$A x = b$$

was rewritten as

(1) $$x = G x + c$$

and an iterative matric equation

(2) $$x^{(n+1)} = G x^{(n)} + c$$

was written for the Jacobi, Gauss-Seidel, and SOR methods. The error associated with the kth iterated vector is defined as

(3) $$\epsilon^{(k)} \equiv x - x^{(n)}$$

The iterated vectors will eventually converge to the solution, x, only if the error vectors converge to zero as k increases toward infinity. A relation between successive iterative errors can be found by subtracting Equation (2) from Equation (1) and substituting Equation (3) into the result.

$$x - x^{(k+1)} = G x - G x^{(k)}$$

$$x - x^{(k+1)} = G\left(x - x^{(k)}\right)$$

$$\epsilon^{(k+1)} = G\,\epsilon^{(k)} \tag{4}$$

We now consider those iterative matrices which have eigenvectors that span the iterative vector space. Those include all real symmetric and some real nonsymmetric matrices. The convergence condition is found by expanding the error vectors in terms of the eigenvectors.

$$\epsilon^{(o)} = \sum_{i=1}^{n} (c_i e_i)$$

$$\epsilon^{(k)} = G^k\,\epsilon^{(o)}$$

$$\epsilon^{(k)} = G^k \sum_{i=1}^{n} (c_i e_i)$$

$$\epsilon^{(k)} = \sum_{i=1}^{n} (c_i G^k e_i) \tag{6}$$

$$\epsilon^{(k)} = \sum_{i=1}^{n} (c_i \lambda_i^k e_i) \tag{7}$$

In order for $\epsilon^{(k)}$ to converge to zero for all c_i and e_i, the following must be true:

$$\lim_{k\to\infty} \lambda_i^{\,k} = 0$$

$$\lambda_i < 1 \quad \text{for all } i$$

$$\rho < 1 \tag{8}$$

This is the convergence condition for iterative matrices that have eigenvectors which span the iterative vector space.

Unfortunately, some iterative matrices are nonsymmetric and have eigenvalues of multiplicity greater than unity. In these cases the eigenvectors might need to be supplimented by principal vectors in order to span the vector space

(5:28-30;8:8,17). *In these cases, some of the terms in* Equations (6) and (7) like

$$c_i G^k e_i = c_i \lambda_i^k e_i \tag{9}$$

are replaced by terms, derived in Appendix B, like

$$c_i G^k t_i = c_i \sum_{i=0}^{i-m} \left[\binom{k}{j} \lambda_m^{k-j} t_{i-j} \right] \tag{10}$$

where

$\binom{k}{j}$ = binomial coefficient = $k!/(j![k-j]!)$ (3:624)

$t_m = e_m$ = an eigenvector associated with the multiple eigenvalue

t_i = principal vectors associated with

$k \geq i-m$, otherwise for $k \leq i-m$, $j=0,k$

It is obvious that $\rho < 1$ is a necessary condition for convergence but it may not be a sufficient condition. For large multiplicities and large eigenvalues, some of the products

$$\binom{k}{j} \lambda_m^{k-j}$$

may not decrease to zero as k increases to infinity.

Note Equation (9) is just a special case of the more general Equation (10) with j=0 signifying that no principal vectors are needed to span the vector space.

If the kth iterated error vector is examined

$$\epsilon^{(k)} = \sum_{i=1}^{n} \left(c_i \lambda_i^k e_i \right)$$

or more generally

$$\epsilon^{(k)} = \sum_{i=1}^{n} \left(c_i \sum_{j=0}^{i-m} \left[\binom{k}{j} \lambda_m^{k-j} t_{i-j} \right] \right)$$

the size of the eigenvalues can be seen to effect the convergence rate. Figure 1 shows that as k increases, λ^k decreases faster for smaller λ . The spectral radius, being the largest eigenvalue in magnitude, gives the slowest decrease in λ^k and is therefore used as an index for convergence rate.

Care must be taken when comparing matric convergence rates by comparing spectral radii. It is quite possible for the matrix with the larger spectral radius initially to converge faster. The matrix with the smaller spectral radius may have much larger secondary eigenvalues than those of the matrix with the larger spectral radius.

$$\begin{bmatrix} .9 & 0 \\ 0 & .1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} .9 \\ .1 \end{bmatrix}, \begin{bmatrix} .81 \\ .01 \end{bmatrix}, \begin{bmatrix} .729 \\ .001 \end{bmatrix}$$

$$\begin{bmatrix} .8 & 0 \\ 0 & .8 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} .8 \\ .8 \end{bmatrix}, \begin{bmatrix} .64 \\ .64 \end{bmatrix}, \begin{bmatrix} .512 \\ .512 \end{bmatrix}$$

Also the choice of initial vector may be linearly independent of the eigenvector associated with the spectral radius.

$$\begin{bmatrix} .9 & 0 \\ 0 & .1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ .1 \end{bmatrix}, \begin{bmatrix} 0 \\ .01 \end{bmatrix}, \begin{bmatrix} 0 \\ .001 \end{bmatrix}$$

$$\begin{bmatrix} .8 & 0 \\ 0 & .8 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ .8 \end{bmatrix}, \begin{bmatrix} 0 \\ .64 \end{bmatrix}, \begin{bmatrix} 0 \\ .512 \end{bmatrix}$$

Comparing two matrices with the same spectral radius, the one with the largest secondary eigenvalue will eventually

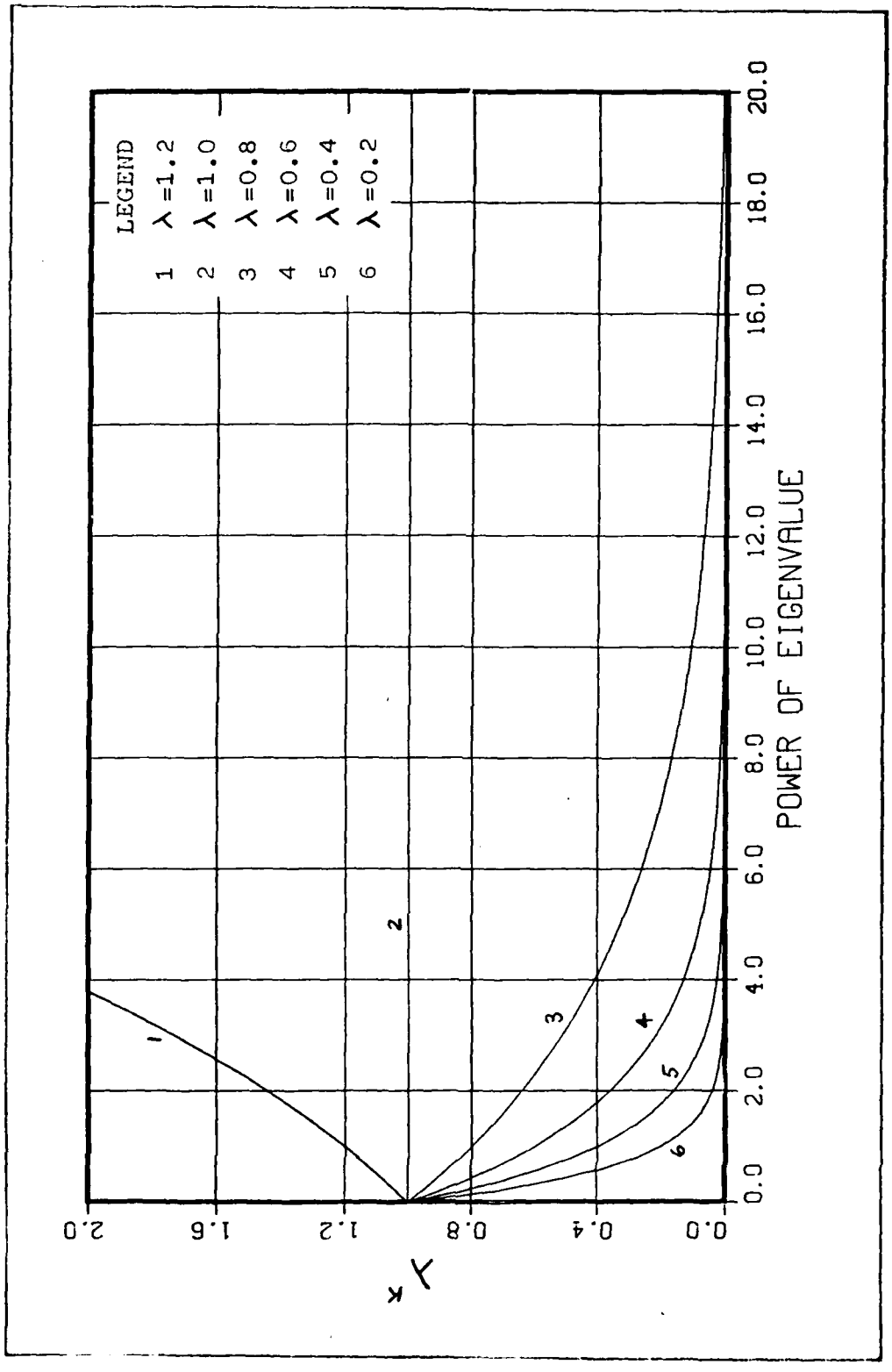


Figure 1. Powers of Eigenvalues

converge slower.

$$\begin{bmatrix} .8 & 0 \\ 0 & .8 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} .8 \\ .8 \end{bmatrix}, \begin{bmatrix} .64 \\ .64 \end{bmatrix}, \begin{bmatrix} .512 \\ .512 \end{bmatrix}$$

$$\begin{bmatrix} .8 & 0 \\ 0 & .4 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} .8 \\ .4 \end{bmatrix}, \begin{bmatrix} .64 \\ .16 \end{bmatrix}, \begin{bmatrix} .512 \\ .064 \end{bmatrix}$$

The following discussion is limited to the matrix problem

$$A x = b$$

where A is real, symmetric, and block tridiagonal (5:45-46). The discussion can be generalized for matrices with other consistent orderings (8).

The Jacobi iteration matrix is real and symmetric and so its eigenvalues are real. Due to the consistent ordering of the elements of matrix A, the nonzero eigenvalues of the Jacobi matrix form positive and negative pairs (8). Also due to the consistent ordering of the elements in matrix A, a special relationship exists between the eigenvalues of the Jacobi and SOR iteration matrices. (Figure 2)

$$(\lambda + w - 1)^2 = \lambda w^2 \mu^2 \qquad (12:243)$$

$$\lambda + w - 1 = \sqrt{(\lambda)}\, w \mu$$

$$\lambda(1) + \lambda^{1/2}(-w\mu) + (w-1) = 0$$

$$\lambda^{1/2} = (w\mu \pm \sqrt{[w^2\mu^2 - 4(w-1)]})/2$$

$$\lambda = (w\mu/2 \pm \sqrt{[w^2\mu^2 - 4(w-1)]/2})^2 \qquad (8)$$

$$= (w\mu/2 \pm \sqrt{[w^2\mu^2/4 - w + 1]})^2$$

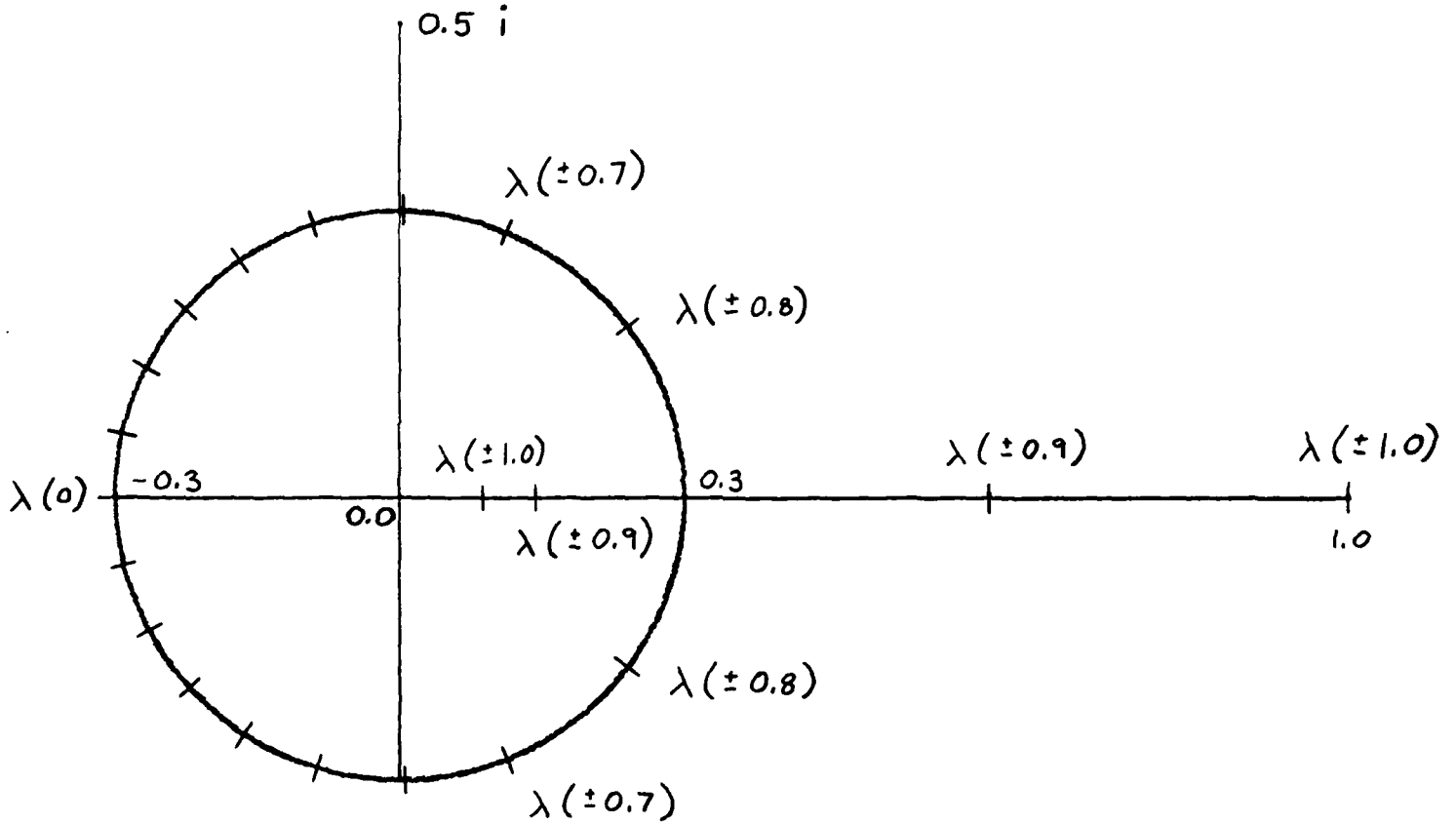


Figure 2. SOR Eigenvalues versus Jacobi Eigenvalues and Acceleration Constant

$$\lambda = (w\mu/2 \pm \sqrt{[(w\mu/2)^2 - w + 1]})^2$$

$$= (w\mu/2)^2 + [(w\mu/2)^2 + 1 - w]$$

$$\pm w\mu \sqrt{[(w\mu/2)^2 + 1 - w]}$$

$$= w^2\mu^2/2 + 1 - w \pm w\mu \sqrt{(w^2\mu^2/4 + 1 - w)}$$

where

μ = Jacobi eigenvalue

w = SOR acceleration constant

λ = SOR eigenvalue

We now note some basic properties of this relationship (8).

1. No $\lambda = 0$

$$(\lambda + w - 1)^2 = \lambda w^2 \mu^2$$

$$\lambda = 0$$

$$(w-1)^2 = 0$$

$$w - 1 = 0$$

$$w = 1$$

but $w > 1$ for SOR.

2. The only negative real eigenvalue is $\lambda = 1 - w$

3. $\lambda = 1 - w$ if and only if $\mu = 0$

$$(\lambda + w - 1)^2 = \lambda w^2 \mu^2$$

$$\mu = 0$$

$$(\lambda + w - 1)^2 = 0$$

$$\lambda + w - 1 = 0$$

$$\lambda = 1 - w$$

and

$$(\lambda + w - 1)^2 = \lambda w^2 \mu^2$$

$$\lambda = 1 - w$$

$$0 = (1 - w) w^2 \mu^2$$

$$\mu \neq 0$$

$$0 = (1 - w) w^2$$

$$w = 1 \quad \text{or} \quad w = 0$$

but $w > 1$ for SOR

4. λ is positive and real if the radical is greater than zero. $|\mu|$ is then $> 2\sqrt{(w-1)}/w$.

$$w^2 \mu^2 / 4 + 1 - w > 0$$

$$w^2 \mu^2 / 4 > w - 1$$

$$\mu^2 > 4 (w-1) / w^2$$

$$|\mu| > 2\sqrt{(w-1)/w}$$

5. $\lambda = w-1$ if and only if the radical equals zero

$$\lambda = w^2\mu^2/2 + 1 - w \pm w\mu\sqrt{(w^2\mu^2/4 + 1 - w)}$$

$$w^2\mu^2/4 + 1 - w = 0$$

$$= w^2\mu^2/2 + 1 - w$$

$$w^2\mu^2/4 + 1 - w = 0$$

$$w^2\mu^2/4 = w - 1$$

$$w^2\mu^2/2 = 2w - 2$$

$$= 2w - 2 + 1 - w$$

$$= w - 1$$

and $|\mu| = 2\sqrt{(w-1)/w}$

$$w^2\mu^2/4 + 1 - w = 0$$

$$w^2\mu^2/4 = w - 1$$

$$\mu^2 = 4(w-1)/w^2$$

$$|\mu| = 2\sqrt{(w-1)/w}$$

6. The number of positive eigenvalues $> w-1$ = the number of positive eigenvalues $< w-1$.

7. λ is complex if and only if the radical is less than zero and $\mu \neq 0$. $|\mu|$ is then $< 2\sqrt{(w-1)/w}$.

$$w^2\mu^2/4 + 1 - w < 0$$

$$w^2\mu^2/4 < w - 1$$

$$\mu^2 < 4(w-1)/w^2$$

$$|\mu| < 2\sqrt{(w-1)/w}$$

8. If λ is complex, the modulus of λ equals w-1.

$$\sqrt{(\lambda)} = w\mu/2 \pm \sqrt{[w^2\mu^2 - 4(w-1)]/2}$$

$$w^2\mu^2 - 4(w-1) < 0$$

$$= w\mu/2 \pm i\sqrt{[4(w-1) - w^2\mu^2]/2}$$

$$|\lambda| = \sqrt{(\lambda^*\lambda)} = [\sqrt{(\lambda)}]^* \sqrt{(\lambda)}$$

$$= w^2\mu^2/4 + [4(w-1) - w^2\mu^2]/4$$

$$= w^2\mu^2/4 + w - 1 - w^2\mu^2/4$$

$$= w - 1$$

The mapping between Jacobi and SOR eigenvalues

$$\lambda = w^2\mu^2/2 + 1 - w \pm w\mu\sqrt{(w^2\mu^2/4 + 1 - w)}$$

gives a pair of unequal SOR eigenvalues for each pair of Jacobi eigenvalues unless the radical equals zero or unless m=0. When the radical equals zero and m≠0, then the Jacobi eigenvalue pair maps into a pair of equal SOR eigenvalues. A Jacobi eigenvalue equal to zero gives a one to one mapping.

A special relationship between eigenvalues of the

Table I

Summary of Jacobi-SOR Eigenvalue Relationships

Jacobi Eigenvalue	Radical	SOR Eigenvalue
—	—	$\lambda \neq 0$
$\lvert\mu\rvert > 2\sqrt{(w-1)}/w$	$r > 0$	$\lambda > 0$
$\lvert\mu\rvert = 2\sqrt{(w-1)}/w$	$r = 0$	$\lambda = w-1 > 0$
$0 \leq \lvert\mu\rvert < 2\sqrt{(w-1)}/w$	$r < 0$	λ = complex , $\lvert\lambda\rvert = w-1$
$\mu = 0$	$r = 1-w < 0$	$\lambda = 1-w < 0$

Jacobi and Gauss-Seidel iteration matrices is obtained as a limiting case of the Jacobi-SOR relationship where w=1

$$\lambda = w^2\mu^2/2 + 1 - w \pm w\mu\sqrt{(w^2\mu^2/4 + 1 - w)}$$

$$w = 1$$

(11) $$\lambda = \mu^2/2 \pm \mu^2/2$$

Note that

1. λ is always real

2. half of the λ's ≥ 0

3. the other half of the λ's $= 0$

Note that the minimum SOR spectral radius with respect to the acceleration constant occurs when the radical equals zero, therefore

$$w^2\mu^2/4 + 1 - w = 0$$

$$w^2\mu^2/4 = w - 1$$

$$\mu^2 = 4(w-1)/w^2$$

$$= 4/w - 4/w^2$$

$$1 - \mu^2 = 4/w^2 - 4/w + 1$$

$$= (2/w - 1)^2$$

$$\sqrt{(1-\mu^2)} = 2/w - 1$$

$$1 + \sqrt{(1-\mu^2)} = 2/w$$

$$w = 2 / [1 + \sqrt{(1 - \mu^2)}]$$

$$w_b = 2 / [1 + \sqrt{(1 - \rho(B)^2)}] \qquad (8;12:243)$$

where

$\rho(B)$ = Jacobi spectral radius

w_b = optimum acceleration constant

Thus, an optimum acceleration constant (Figure 3) exists that minimizes the spectral radius and so maximizes convergence.

The spectral radius corresponding to the optimum acceleration constant can be found by recalling that when the radical equals zero, $\lambda = w-1$.

$$\rho(SOR) = w_b - 1$$

$$w_b = 2/(1 + \sqrt{[1 - \rho(B)^2]}) - 1$$

$$= 2/(1 + \sqrt{[1 - \rho(B)^2]}) - 1 \qquad (12)$$

Figure 4 compares the Jacobi, Gauss-Seidel, and optimum SOR spectral radii.

Again, due to the consistent ordering of the elements in matrix A, a special relationship exists between the eigenvectors of the Jacobi and SOR iteration matrices (8).

$$y = E[\sqrt{(\lambda)}]\, v$$

where

y = SOR eigenvector

λ = SOR eigenvalue

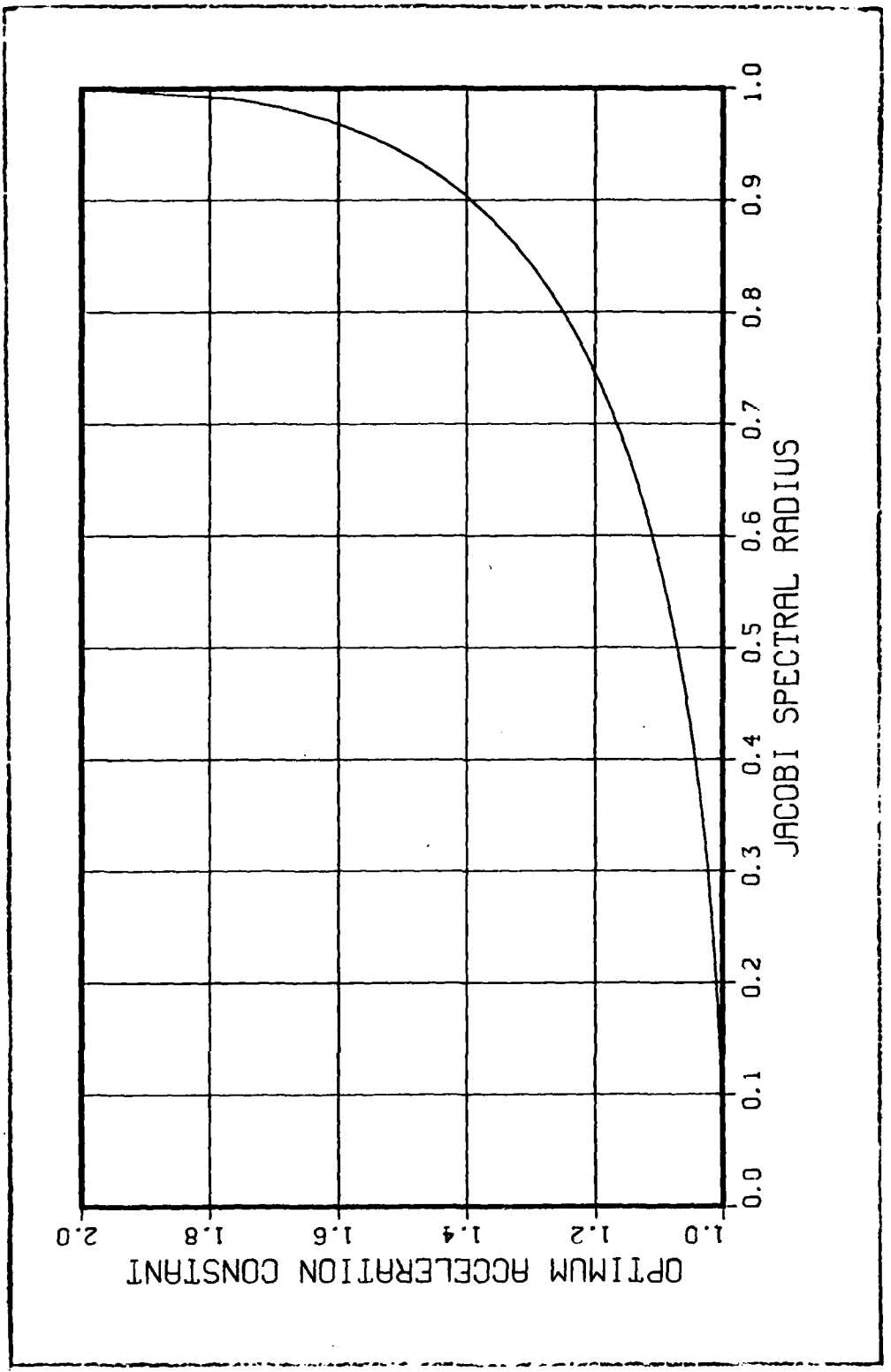


Figure 3. Optimum Acceleration Constant

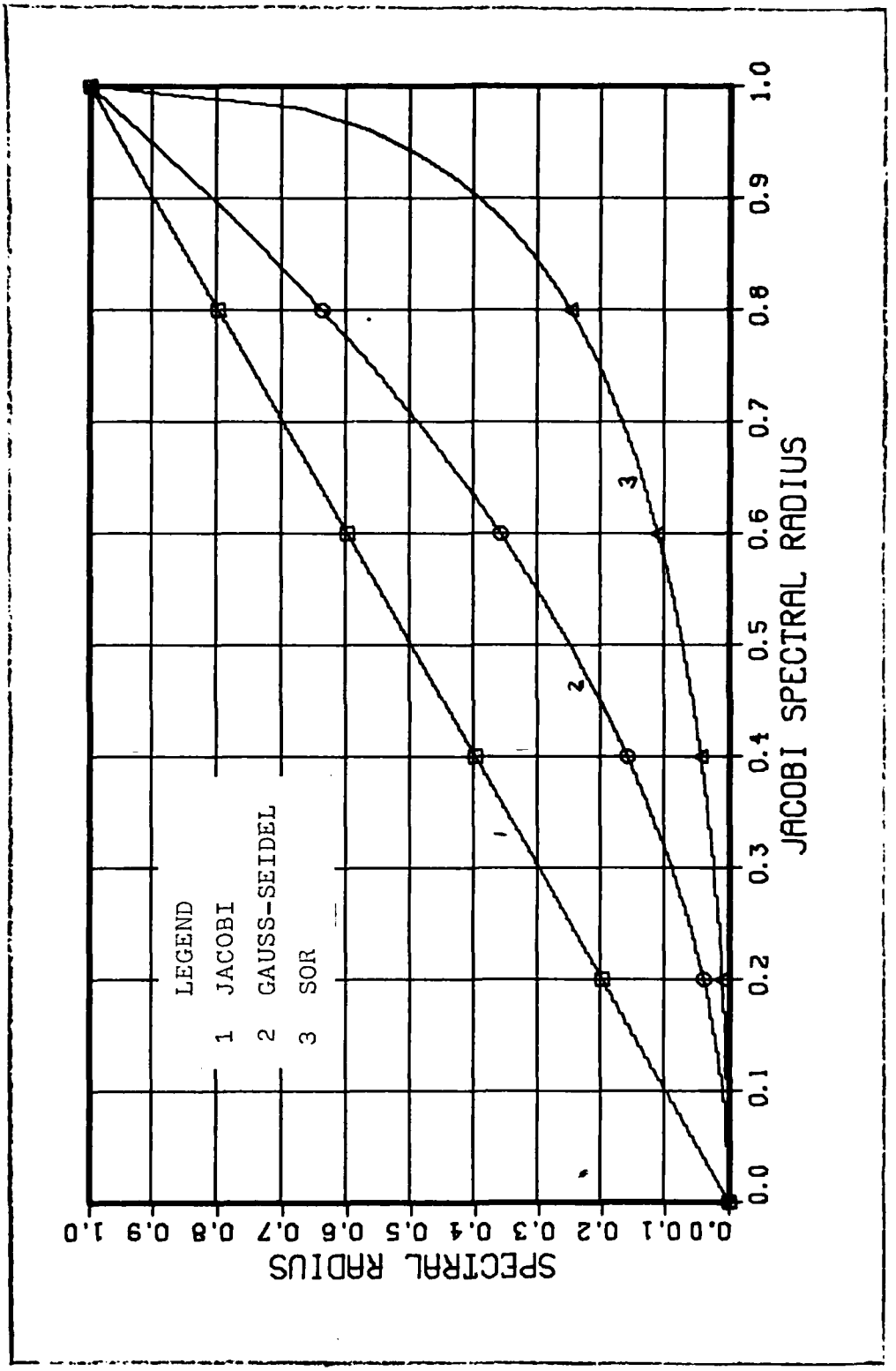


Figure 4. Spectral Radius Comparison

v = Jacobi eigenvector associated with the Jacobi eigenvalue $\mu \geq 0$ that maps into λ .

$$E[x] = \begin{bmatrix} I_1 & & \\ & x\, I_2 & \\ & & x^2 I_3 \\ & & & \ddots \end{bmatrix}$$

a diagonal matrix in which the identity submatrices I_j are of the same order as the diagonal submatrices of the block tridiagonal matrix A (5:46).

If the radical $\neq 0$, then the Jacobi eigenvectors will give an equal number of linearly independent SOR eigenvectors. But if the radical = 0, then the Jacobi eigenvectors will yield only half as many linearly independent SOR eigenvectors. Principal vectors will have to be added in order to span the vector space (8).

$$p = F[\sqrt{(w-1)}]\, v \,/\, \sqrt{(w-1)}$$

where

p = principal vector

v = Jacobi eigenvector associated with the that maps into $\lambda = w-1$

$F[x]$ = the diagonal matrix with elements $F_j = dE_j/dx$

The special relationship between eigenvectors of the Jacobi and Gauss-Seidel iteration matrices is obtained as a limiting case of the Jacobi-SOR relation where w=1. The relation is unchanged (8)

$$y = E[\sqrt{(\lambda)}]\, v$$

where

y = Gauss-Seidel eigenvector

λ = Gauss-Seidel eigenvalue

v = Jacobi eigenvector associated with the Jacobi eigenvalue $\mu \geq 0$ that maps into λ .

$E[x]$ = Same as previously described

Note that Gauss-Seidel eigenvectors are real and span the vector space.

Generally the Gauss-Seidel and SOR matrices are not symmetric so that these special relationships are welcome.

Norms (6:171-173,176)

It is often convenient to measure the size of a vector or a square matrix by assigning a scalar number to it. A norm does just that. This section introduces the concept of norm which will be used later to measure iterative vector errors. The norm will be defined and some useful relationships given.

A norm is symbolized by placing $\| \ \|$ around the vector or matrix symbol. Properties that define a norm are analogous to properties that define an absolute value. In fact, an absolute value is a norm.

$$\|A\| \geq 0$$

$\|A\| = 0$ if and only if A=0

$$\|\alpha A\| = |\alpha| \, \|A\|$$

$$\|A+B\| \leq \|A\| + \|B\|$$

A matrix norm must also have the following property

$$\|A\,B\| \leq \|A\|\,\|B\|$$

A vector n-norm can be defined as

$$\|V\|_n \equiv \sqrt[n]{\left(\sum_i |V_i|^n\right)}$$

where

V = vector

V_i = ith vector element

Commonly used vector norms are the one-norm, the two-norm or euclidean-norm, and the infinite-norm, respectively

$$\|V\|_1 = \sum_i |V_i|$$

$$\|V\|_2 = \sqrt{\left(\sum_i |V_i|^2\right)}$$

$$\|V\|_\infty = \max_i |V_i|$$

Vector norms can be ordered as follows

$$\|V\|_1 \geq \|V\|_2 \geq \|V\|_3 \geq \cdots \geq \|V\|_\infty$$

A matric n-norm can be defined as

$$\|M\|_n \equiv \max_V \left(\|M\,V\|_n / \|V\|_n\right)$$

Matric norms are generally difficult to calculate. Commonly used matric norms are the one-norm and the infinite

norm.

$$\|M\|_1 = \max_j \left(\sum_i |m_{ij}| \right)$$
$$\|M\|_\infty = \max_i \left(\sum_j |m_{ij}| \right)$$

where

m_{ij} = the matric element of the ith row and jth column.

If a matrix is symmetric, then

$$\|M\|_1 = \|M\|_\infty$$

The spectral radius is less than or equal to all matric n-norms (12:237).

$$\rho \leq \|M\|_n \qquad \text{for all } n$$

Conte and DeBoor state without proof (6:228), that for any $\varepsilon > 0$ a matric norm exists such that

$$\|M\| \leq \rho(M) + \varepsilon$$

The triangle inequality property of a norm can be extended as follows

$$\|A\| + \|B\| \geq \|A \pm B\| \geq |\,\|A\| - \|B\|\,| \geq \|A\| - \|B\|$$

One last inequality can be written

$$\|M\|\,\|v\| \geq \|Mv\| \geq \|v\| / \|M^{-1}\|$$

Iterative Error Limits

The concept of contractive matrices is introduced to aid in measuring the error between the matric solution and the iterated solution. The contractive constant and displacement vector will be introduced to estimate the error vector.

Recall the implicit matric equation

$$A x = b$$

which was rewritten as (12:236)

$$x = G x + c$$

so that an iterated solution could be attempted using

$$x^{(n+1)} = G x^{(n)} + c$$

where

X = matric solution

$x^{(n)}$ = nth iterated solution

G = iteration matrix

c = known constant vector

The error vector was defined as

$$e^{(n)} \equiv x - x^{(n)}$$

and the displacement vector was defined as

$$d^{(n)} \equiv x^{(n+1)} - x^{(n)}$$

Simple relationships exist between the iteration matrix, the error vector, and the displacement vector.

$$x = Gx + c$$

$$x^{(n+1)} = Gx^{(n)} + c$$

$$x - x^{(n+1)} = Gx - Gx^{(n)}$$

$$= G(x - x^{(n)})$$

$$e^{(n)} \equiv x - x^{(n)}$$

(1) $$e^{(n+1)} = G\,e^{(n)}$$ (8;12:236)

$$x^{(n+2)} = Gx^{(n+1)} + c$$

$$x^{(n+1)} = Gx^{(n)} + c$$

$$x^{(n+2)} - x^{(n+1)} = Gx^{(n+1)} - Gx^{(n)}$$

$$= G(x^{(n+1)} - x^{(n)})$$

$$d^{(n)} \equiv x^{(n+1)} - x^{(n)}$$

(2) $$d^{(n+1)} = G\,d^{(n)}$$ (8)

$$d^{(n)} = G\,d^{(n-1)}$$

$$d^{(n-1)} = x^{(n)} - x^{(n-1)}$$

$$= G(x^{(n)} - x^{(n-1)})$$

$$e^{(n-1)} = x - x^{(n-1)}$$

$$e^{(n)} = x - x^{(n)}$$

$$e^{(n-1)} - e^{(n)} = x^{(n)} - x^{(n-1)}$$

$$d^{(n)} = G\left(e^{(n-1)} - e^{(n)}\right)$$

$$= G e^{(n-1)} - G e^{(n)}$$

$$e^{(n)} = G e^{(n-1)}$$

$$= e^{(n)} - G e^{(n)}$$

$$= (I - G)\, e^{(n)}$$

(3) $$e^{(n)} = (I - G)^{-1} d^{(n)} \qquad (8)$$

This might suggest that by knowing the displacement we might be able to find the error, or at least an approximation or upper limit.

If our iterative matric function

$$g(z) = G z + c$$

is contractive, then for all z, a contractive constant, K, exists such that

$$0 \leq K < 1$$

and

$$\|g(x) - g(y)\| \leq K \|x - y\|$$

for some matric norm (6:223). This defines a contractive function. Simple relationships between the displacement and error vectors are found to exist by using this con-

tractive property.

$$x = G x + c$$

$$x^{(n+1)} = G x^{(n)} + c$$

$$\| x - x^{(n+1)} \| \leq K \| x - x^{(n)} \|$$

$$e^{(n)} \equiv x - x^{(n)}$$

$$\| e^{(n+1)} \| \leq K \| e^{(n)} \| \tag{4}$$

$$x^{(n+2)} = G x^{(n+1)} + c$$

$$x^{(n+1)} = G x^{(n)} + c$$

$$\| x^{(n+2)} - x^{(n+1)} \| \leq K \| x^{(n+1)} - x^{(n)} \|$$

$$d^{(n)} \equiv x^{(n+1)} - x^{(n)}$$

$$\| d^{(n+1)} \| \leq K \| d^{(n)} \| \tag{5}$$

$$x = G x + c$$

$$x^{(n+1)} = G x^{(n)} + c$$

$$\| x - x^{(n)} \| K \geq \| x - x^{(n+1)} \|$$

$$= \| x - x^{(n)} + x^{(n)} - x^{(n+1)} \|$$

$$= \| (x - x^{(n)}) - (x^{(n+1)} - x^{(n)}) \|$$

$$\geq \| x - x^{(n)} \| - \| x^{(n+1)} - x^{(n)} \|$$

$$\| x^{(n+1)} - x^{(n)} \| \geq \| x - x^{(n)} \| - \| x - x^{(n)} \| K$$

$$\| x^{(n+1)} - x^{(n)} \| \geq \| x - x^{(n)} \| \ (1-K)$$

$$d^{(n)} = x^{(n+1)} - x^{(n)}$$

$$e^{(n)} = x - x^{(n)}$$

$$\| d^{(n)} \| \geq \| e^{(n)} \| \ (1-K)$$

$$1 > K$$

$$1 - K > 0$$

$$\| e^{(n)} \| \leq \| d^{(n)} \| \ / \ (1-K) \tag{6}$$

These are analogous to the previous relationships.

A variety of relationships between errors and displacements of differing iterations can be obtained by applying either or both Equations (4) and (5) to (6) successively. For example,

$$\| e^{(n)} \| \leq \| d^{(n-1)} \| \ K \ / \ (1-K) \tag{7}$$ (6:223)

$$\| e^{(n)} \| \leq \| d^{(0)} \| \ K^n \ / \ (1-K) \tag{8}$$ (6:223)

The same can be done with Equations (1), (2), and (3)

$$e^{(n)} = (I-G)^{-1} \ G \ d^{(n-1)}$$

$$e^{(n)} = (I-G)^{-1} \ G^n \ d^{(0)}$$

Note that

$$\| e^{(n)} \| \leq \| d^{(n)} \| / (1-K) \leq \| d^{(n-1)} \| \ K / (1-K) \leq \| d^{(0)} \| \ K^n / (1-K)$$

How is it known that a function $g(z)$ is contractive?

How is it known that a contractive constant less than unity exists? Assume the matric function is contractive.

$$\| g(x) - g(y) \| \leq K \| x - y \|$$

$$g(x) = G x + c$$

$$g(y) = G y + c$$

$$g(x) - g(y) = G x - G y$$

$$= G (x - y)$$

$$v \equiv x - y$$

$$= G v$$

$$\| G v \| \leq K \| v \| \qquad \text{for all } v$$

$$\| G v \| / \| v \| \leq K < 1 \qquad \text{for all } v$$

$$\max_v \| G v \| / \| v \| \leq K < 1$$

$$\| G \| \equiv \max_v \| G v \| / \| v \|$$

$$\| G \| \leq K < 1$$

Note that $\| G \|$ is always an acceptable K if $\| G \|$ is less than unity.

$$g(x) - g(y) = G (x - y)$$

$$\| g(x) - g(y) \| = \| G (x - y) \|$$

$$\leq \| G \| \| x - y \|$$

Thus our iterative matric function is contractive if and only if some matric norm of G is less than unity (6:225).

$$\|G\| < 1$$

Recall, from the section on norms, that a matric norm exists such that

$$\rho(G) \leq \|G\| \leq \rho(G) + \varepsilon$$

where

$\rho(G)$ = spectral radius of the matrix G

ε = any number greater than zero

Thus a matric function is contractive if and only if the spectral radius is less than unity (6:228)

$$\rho(G) < 1$$

This is the same condition for convergence.

Normally one-norms and infinite-norms are used for analysis because they are easy to compute and they best represent the type data to be analyzed. Unfortunately, the contractive relationships are valid only for those vector norms associated with matric norms less than unity. Recall, from the section on norms, that if the magnitude of any matric element is greater than unity, then the infinite-norm and one-norm are also greater than unity.

Even though the relationships

(4) $$\|e^{(n+1)}\| \leq K \|e^{(n)}\|$$

(5) $$\|d^{(n+1)}\| \leq K \|d^{(n)}\|$$

(6) $$\|e^{(n)}\| \leq \|d^{(n)}\| / (1-K)$$

(7) $$\|e^{(n)}\| \leq \|d^{(n-1)}\| \, K / (1-K)$$

may be invalid for K limited to $1 > K \geq 0$, extended domains of K exist where the relationships are always valid, whether the matric function is contractive or not.

Recall Equation (1)

$$e^{(n+1)} = G e^{(n)}$$

$$\|e^{(n+1)}\| = \|G e^{(n)}\|$$

$$\leq \|G\| \, \|e^{(n)}\|$$

Comparing this with Equation (4) shows that

$$0 \leq K \leq \|G\|$$

is the extended domain of K for Equation (4). Similarly, recall Equation (2)

$$d^{(n+1)} = G d^{(n)}$$

$$\|d^{(n+1)}\| = \|G d^{(n)}\|$$

$$\leq \|G\| \, \|d^{(n)}\|$$

Comparing this with Equation (5) shows that

$$0 \leq K \leq \|G\|$$

is also the extended domain of K for Equation (5). By inspection, the extended domain of K for Equation (6) is (Figure 5)

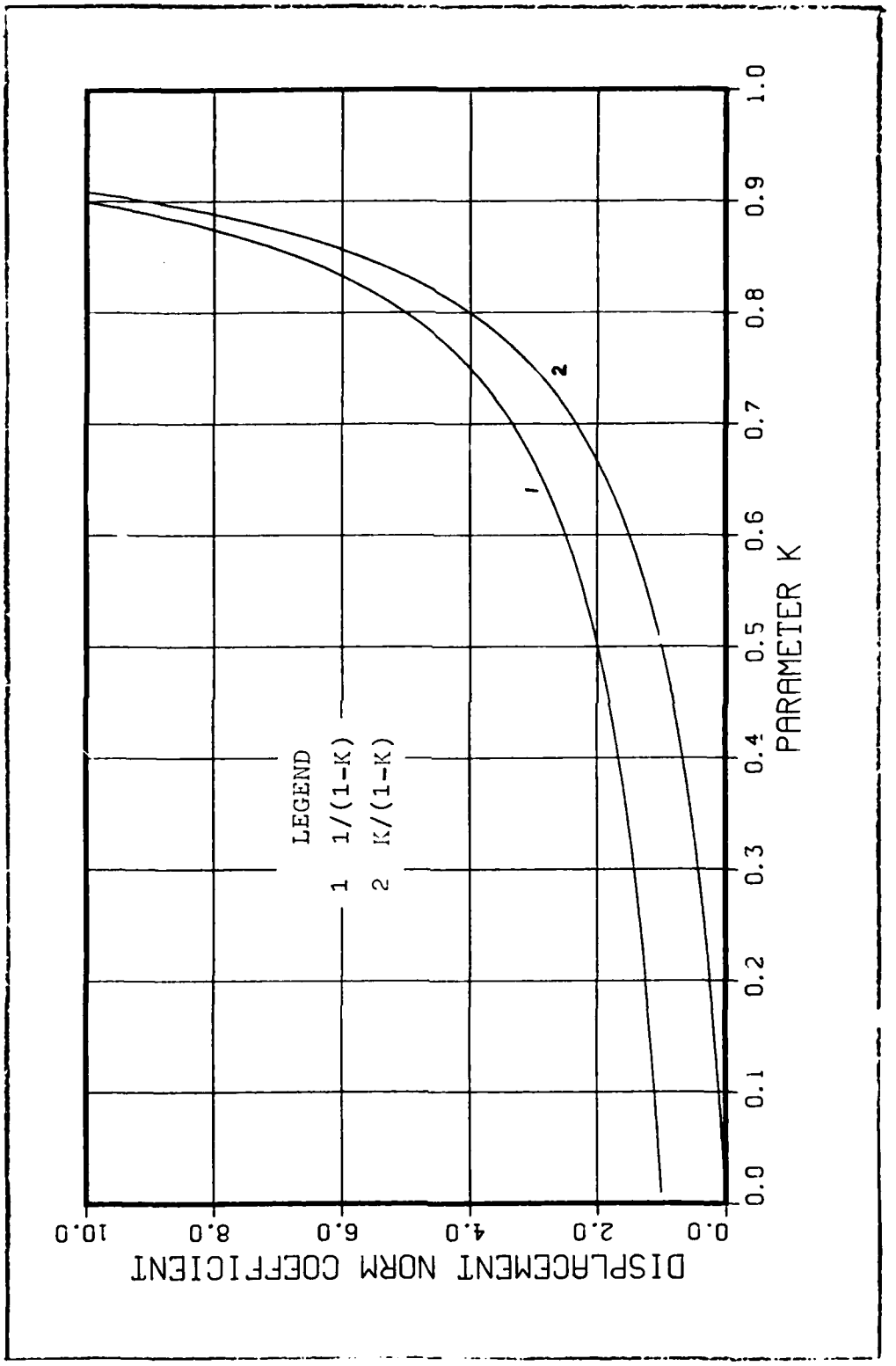


Figure 5. The Extended Domain of Parameter K

$$-\infty \leq K < 1$$

By inspection, the domain of K for Equation (7) needs no extension (Figure 5)

$$0 \leq K < 1$$

Finally we analyze how these four relationships behave in the limit as the number of iterations increases toward infinity. This is the region where our iterative solution converges to our matric solution. Note that the following derivations assume that the eigenvectors span the vector space and that the principal eigenvalue is real and does not have a negative counterpart

$$\lambda_1 \neq -\lambda_i \qquad \text{for any } i$$

Recall the sections on spectral radius and iterative matric convergence for difficulties that can occur.

First we derive the $\lim_{n\to\infty} \|e^{(n+1)}\| / \|e^{(n)}\|$. Expanding the initial error in terms of the eigenvectors, v, of the matrix, G (12:238)

$$e^{(0)} = \sum_{i=1}^{m} (c_i v_i)$$

$$e^{(n)} = G^n e^{(0)}$$

$$= G^n \sum_{i=1}^{m} (c_i v_i)$$

$$= \sum_{i=1}^{m} (c_i G^n v_i)$$

$$= \sum_{i=1}^{m} (c_i \lambda_i^n v_i)$$

$$= \lambda_1^n \left[\sum_{j=1}^{s} (c_j v_j) + \sum_{i=s+1}^{m} (c_i [\lambda_i/\lambda_1]^n v_i) \right]$$

where λ_1 is of multiplicity s.

$$\lambda_i < \lambda_1$$

$$|\lambda_i/\lambda_1| < 1$$

$$\lim_{n\to\infty} [\lambda_i/\lambda_1]^n = 0$$

$$\lim_{n\to\infty} e^{(n)} = \lambda_1^n \sum_{j=1}^{s} (c_j v_j)$$

which is an eigenvector.

(10) $$\lim_{n\to\infty} e^{(n+1)} = \lim_{n\to\infty} e^{(n)} \lambda_1$$

Using the quotient method described in the section on spectral radius

$$\lim_{n\to\infty} w^T e^{(n+1)} / (w^T e^{(n)}) = \lambda_1$$

or using norms

$$\lim_{n\to\infty} \|e^{(n+1)}\| = \lim_{n\to\infty} \|\lambda_1 e^{(n)}\|$$

$$= |\lambda_1| \lim_{n\to\infty} \|e^{(n)}\|$$

(11) $$\lim_{n\to\infty} \|e^{(n+1)}\| / \|e^{(n)}\| = |\lambda_1|$$

$$\equiv \rho(G)$$

Similarly (10:241)

$$\lim_{n\to\infty} d^{(n+1)} = \lim_{n\to\infty} d^{(n)} \lambda_1$$

$$\lim_{n\to\infty} w^T d^{(n+1)} / (w^T d^{(n)}) = \lambda_1$$

(12) $$\lim_{n\to\infty} \|d^{(n+1)}\| \,/\, \|d^{(n)}\| = |\lambda_1|$$

Next we derive $\lim_{n\to\infty} \|e^{(n)}\| \,/\, \|d^{(n)}\|$ (8;12:240)

$$e^{(n)} = x - x^{(n)}$$

$$e^{(n+1)} = x - x^{(n+1)}$$

$$e^{(n)} - e^{(n+1)} = x^{(n+1)} - x^{(n)}$$

$$d^{(n)} \equiv x^{(n+1)} - x^{(n)}$$

$$e^{(n)} - e^{(n+1)} = d^{(n)}$$

$$\lim_{n\to\infty} \left(e^{(n)} - e^{(n+1)}\right) = \lim_{n\to\infty} d^{(n)}$$

$$\lim_{n\to\infty} \left(e^{(n)} - \lambda_1 e^{(n)}\right) = \lim_{n\to\infty} d^{(n)}$$

$$\lim_{n\to\infty} e^{(n)}\left(1 - \lambda_1\right) = \lim_{n\to\infty} d^{(n)}$$

$$\lim_{n\to\infty} \|e^{(n)}(1 - \lambda_1)\| = \lim_{n\to\infty} \|d^{(n)}\|$$

$$\lim_{n\to\infty} \|e^{(n)}\| \,|1 - \lambda_1| = \lim_{n\to\infty} \|d^{(n)}\|$$

$$\lim_{n\to\infty} \|e^{(n)}\| \,/\, \|d^{(n)}\| = 1 \,/\, |1 - \lambda_1|$$

$$\lambda_1 < 1$$

(13) $$\lim_{n\to\infty} \|e^{(n)}\| \,/\, \|d^{(n)}\| = 1 \,/\, (1 - \lambda_1)$$

Note, the difficulty of having $\lambda_1 = -\lambda_i$ is apparent from this last equation. Two limits would exist, which is nonsense.

Similarly

$$\lim_{n\to\infty} e^{(n)}(1-\lambda_1) = \lim_{n\to\infty} d^{(n-1)}\lambda_1$$

$$\lim_{n\to\infty} \|e^{(n)}\| / \|d^{(n-1)}\| = |\lambda_1| / (1-\lambda_1) \tag{14}$$

Note that $\lambda_1 < 0$ can not be a K in the limit because of the absolute value. An equivalent K can be derived. (Figure 6)

$$K/(1-K) = |\lambda_1| / (1-\lambda_1)$$

$$K(1-\lambda_1) = |\lambda_1|(1-K)$$

$$K - K\lambda_1 = |\lambda_1| - K|\lambda_1|$$

$$K - K\lambda_1 + K|\lambda_1| = |\lambda_1|$$

$$K(1-\lambda_1 + |\lambda_1|) = |\lambda_1|$$

$$K = |\lambda_1| / (1-\lambda_1 + |\lambda_1|) \tag{15}$$

If the principal eigenvalue is negative

$$\lambda_1 < 0$$

$$\lambda_1 = -|\lambda_1|$$

$$K = |\lambda_1| / (1 + |\lambda_1| + |\lambda_1|)$$

$$= |\lambda_1| / (1 + 2|\lambda_1|) \tag{16}$$

If the principal eigenvalue is positive

$$\lambda_1 > 0$$

$$\lambda_1 = |\lambda_1|$$

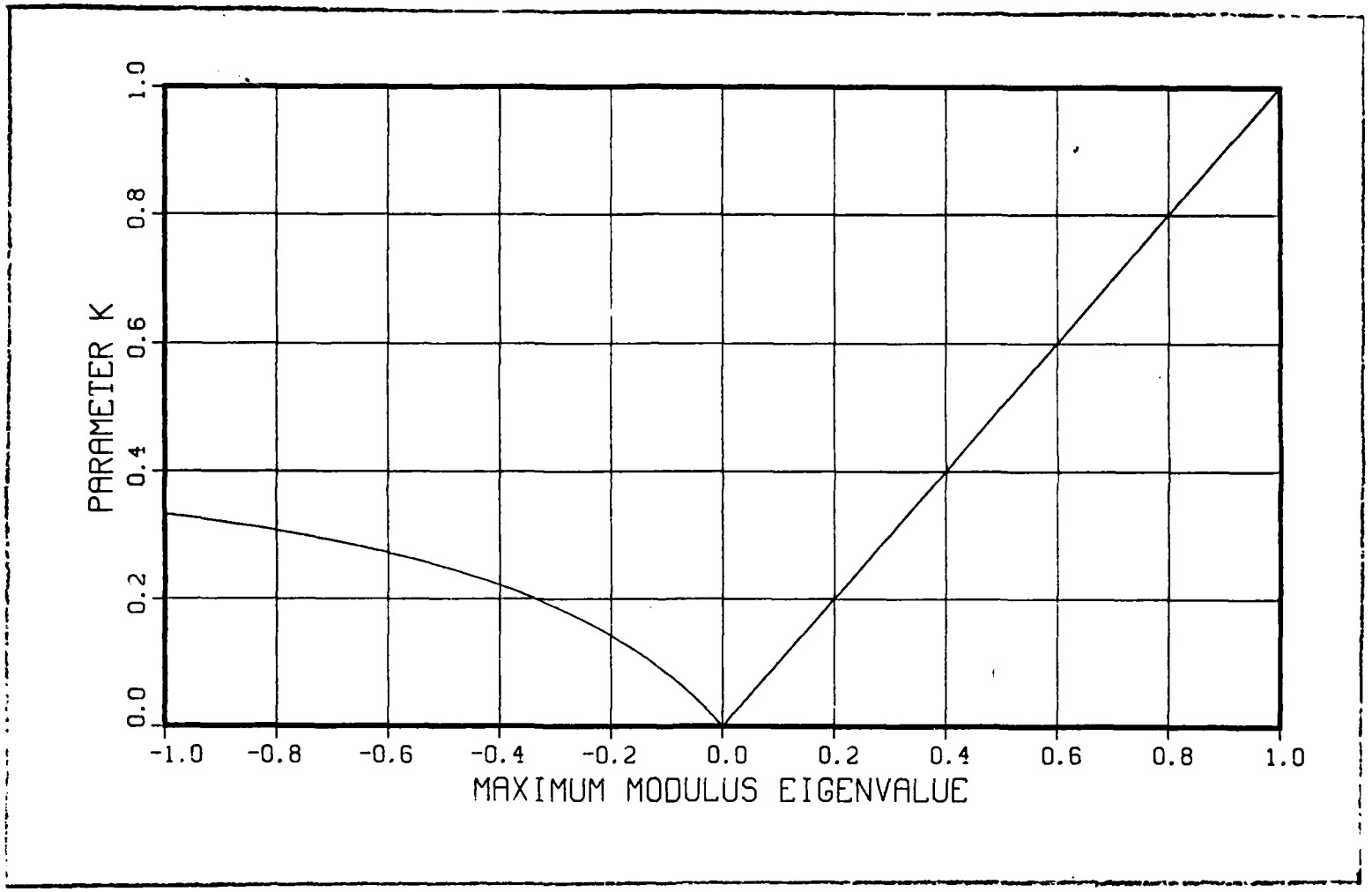


Figure 6. The K that satisfies $\lim_{n \to \infty} \|e^{(n)}\| / \|d^{(n-1)}\| = K/(1-K)$

$$K = |\lambda_1| / (1 - |\lambda_1| + |\lambda_1|)$$

$$= |\lambda_1|$$

as expected.

III. Procedure

Sample Problem

We now pose the sample problem that will be the subject of experimentation. A parabolic two dimensional partial differential equation was written to represent transient heat conduction

$$(1/K)\,\partial T/\partial t = \partial^2 T/\partial x^2 + \partial^2 T/\partial y^2$$

It was then approximated using a full implicit form of differencing

$$(1/K)\,\Delta_t\, T_{i,j,n}\,/h_t$$
$$= \delta_x^2\, T_{i,j,n+1}\,/h_x^2 + \delta_y^2\, T_{i,j,n+1}\,/h_y^2$$

which was then expanded

$$(T_{i,j,n+1} - T_{i,j,n})\,/(h_t\, K)$$
$$= (T_{i+1,j,n+1} - 2\,T_{i,j,n+1} + T_{i-1,j,n+1})\,/h_x^2$$
$$+ (T_{i,j+1,n+1} - 2\,T_{i,j,n+1} + T_{i,j-1,n+1})\,/h_y^2$$

terms collected with unknowns on left side

$$T_{i,j,n} = T_{i,j,n+1}\,(1 + 2h_t K/h_x^2 + 2h_t K/h_y^2)$$
$$+ T_{i+1,j,n+1}\,(-h_t K/h_x^2) + T_{i-1,j,n+1}\,(-h_t K/h_x^2)$$
$$+ T_{i,j+1,n+1}\,(-h_t K/h_y^2) + T_{i,j-1,n+1}\,(-h_t K/h_y^2)$$

and constants defined

$$T_{i,j,n} = a\; T_{i-1,j,n+1}$$

$$+ b \quad T_{i,j-1,n+1}$$

$$+ c \quad T_{i,j,n+1}$$

$$+ b \quad T_{i,j+1,n+1}$$

$$+ a \quad T_{i+1,j,n+1}$$

where

$$a = -K h_t / h_x^2$$

$$b = -K h_t / h_y^2$$

$$c = 1 + 2 h_t K / h_x^2 + 2 h_t K / h_y^2$$

The computational molecule (12:12) that represents the relationships between variables of the difference equation is depicted in Figure 7. The spacial subscripts must be combined before the function T can be represented by a vector. The interior nodes are equated to sequential vector elements. If the spacial domain is rectangular, the subscript i is incremented through its entire range while for each increment of i, the subscript j is incremented through its entire range. The resulting computational molecule is depicted in Figure 8.

This sequencing of interior nodes results in an implicit matric equation

$$A x = b$$

with the matrix, A, being block tridiagonal. The submatrices on the diagonal will be tridiagonal with diagonal elements equal to c and off diagonal elements equal to b. The submatrices on the off diagonal will be diagonal with

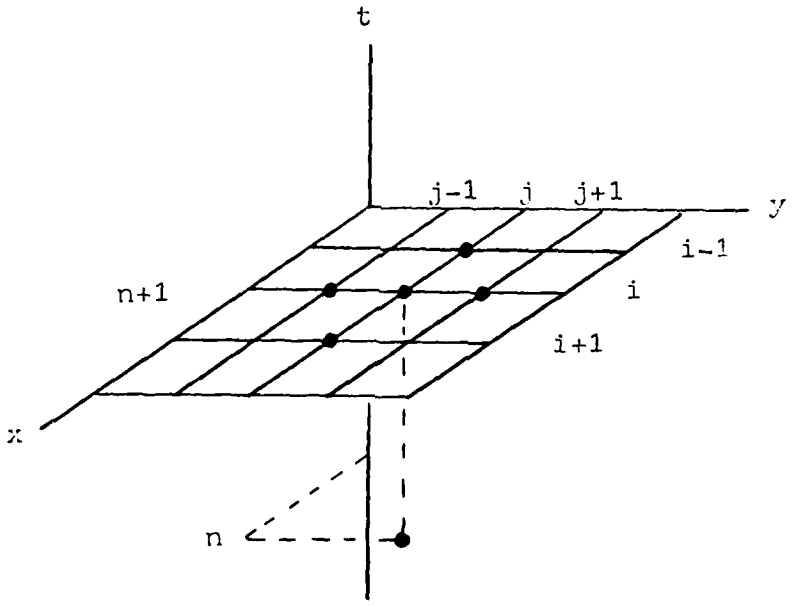


Figure 7. Computational Molecule Using Two Spacial Indices

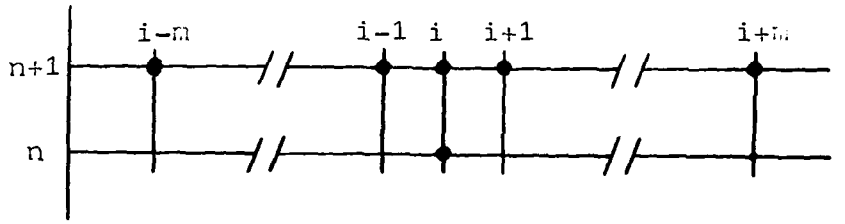


Figure 8. Computational Molecule Using One Spacial Index

diagonal elements equal to a. For example, a rectangular spacial domain with two interior nodes on a side will result in

$$A = \begin{bmatrix} c & b & a & 0 \\ b & c & 0 & a \\ a & 0 & c & b \\ 0 & a & b & c \end{bmatrix}$$

Appendix C contains the analytic solution to the 4 by 4 implicit matric equation, contains the Jacobi, Gauss-Seidel, and SOR 4 by 4 iterative matric equations and the associated eigenvalues. This information was for computer code validation.

It was necessary to pick initial and boundary conditions that resulted in a partial differential equation that could be solved analytically. This physical solution could later be compared to matric solutions, x, and to iterated solutions, $x^{(n)}$.

The analytic solution of the parabolic partial differential equation

$$\partial^2 T/\partial x^2 + \partial^2 T/\partial y^2 = (1/K)\,\partial T/\partial t$$

over the spacial and time domains

$$0 \leq x \leq \pi$$

$$0 \leq y \leq \pi$$

$$0 \leq t$$

with homogenious Dirichlet boundary conditions (1:333)

$$T(0, y, t) = T(\pi, y, t) = 0$$

$$T(x,o,t) = T(x,\pi,t) = 0$$

and an initial condition

$$T(x,y,o) = \sin(x)\sin(y)$$

is equal to

$$T(x,y,t) = \sin(x)\sin(y)\, e^{-2Kt}$$

The derivation and validation is contained in Appendix D.

Being that the boundary conditions are homogeneous, the vector b of the implicit matric equation

$$Ax = b$$

is equal to the initial conditions. The homogeneous boundary conditions contribute only terms equal to zero.

Computer Codes

Two computer codes were designed to measure the iterative matric spectral radius as a function of the density of nodes over the spacial variable domain. The Jacobi spectral radius was first determined using the power method, then the Gauss-Seidel and SOR spectral radii were calculated from Equations (11) and (12). The first code determines the spectral radius using a quotient of iterative vectors separated by one iteration (Appendix E), the second uses a quotient of iterated vectors separated by two iterations (Appendix F). The second code was run to make certain that truncation error forced the two maximum modulus Jacobi eigenvalues to have slightly differing magitudes.

One computer code (Appendix G) was designed to iterate sample problem solutions by using successive over-relaxation. Analytic matric solutions that corresponded to physical

solutions could only be calculated for the low nodal densities, 3 by 3, 4 by 4, and 5 by 5. Matric solutions for higher nodal densities were not correlated with physical solutions. In these cases an initial iterative vector was calculated that would converge to a preplanned matric solution by solving the matric equation in reverse.

$$A x = b$$

where

x = preplanned matric solution

b = calculated initial iteration vector

In this way the error between the iterated vector and the matric solution was easy to calculate.

Five sets of data were generated for later comparison. The first set of data contains the physical and matric solutions and, for each iteration, an iterated solution. The errors between each pair of solutions is also given. The second set contains displacement norm ratios and error norm ratios which can be compared to their limiting value, the spectral radius. The third set contains the values of the parameter that is required to make each error approximation an equality.

$$\|e^{(n)}\| = \|d^{(n)}\| / (1-K)$$

$$\|e^{(n)}\| (1-K) = \|d^{(n)}\|$$

$$(1-K) = \|d^{(n)}\| / \|e^{(n)}\|$$

$$K = 1 - \|d^{(n)}\| / \|e^{(n)}\|$$

$$\|e^{(n)}\| = \|d^{(n-1)}\| \; K / (1-K)$$

$$\|e^{(n)}\| \; (1-K) = \|d^{(n-1)}\| \; K$$

$$\|e^{(n)}\| = (\|d^{(n-1)}\| + \|e^{(n)}\|) K$$

$$K = \|e^{(n)}\| / (\|d^{(n-1)}\| + \|e^{(n)}\|)$$

$$K = 1 / (\|d^{(n-1)}\| / \|e^{(n)}\| + 1)$$

They will reveal how the parameter varies from iteration to iteration and what bounds exist, if any. The fourth set of data contains the error norm to displacement norm ratios actually observed for two of the error approximation methods.

$$\|e^{(n)}\| = \|d^{(n)}\| / (1-K)$$

$$\|e^{(n)}\| / \|d^{(n)}\| = 1 / (1-K)$$

$$\|e^{(n)}\| = \|d^{(n-1)}\| \; K / (1-K)$$

$$\|e^{(n)}\| / \|d^{(n-1)}\| = K / (1-K)$$

They will reveal how the ratio varies from iteration to iteration and what bounds exist, if any. The fifth set of data contains approximations to the error between iterative and matric solution using the following three error methods.

$$\|e^{(n)}\| \leq \|d^{(n)}\| / (1-K)$$

$$\|e^{(n)}\| \leq \|d^{(n-1)}\| \; K / (1-K)$$

$$\|e^{(n)}\| \leq \|d^{(0)}\| \; K^n / (1-K)$$

where the parameter K which satisfies the inequality is approximated by the following six scalar quantities

$$\| d^{(n+1)} \| / \| d^{(n)} \| \simeq K$$

$$\| d^{(n)} \| / \| d^{(n-1)} \| \simeq K$$

$$\rho \quad \text{(Jacobi)} \simeq K$$

$$\rho \quad \text{(Gauss-Seidel)} \simeq K$$

$$\rho \quad \text{(SOR)} \simeq K$$

$$0 \simeq K$$

IV Numerical Results

Spectral Radius Versus Nodal Density

This next topic concerns spectral radius of the iterative matrix versus the spacial node density and the size of time interval between solutions of the implicit matric equation.

Figure 9 depicts the Jacobi spectral radius as a function of nodal density in a square spacial domain. Figure 10 depicts the corresponding SOR optimum spectral radius. Figures 9 and 10 show that the spectral radius increases with increasing nodal density.

Tables II, III, IV, and V list the Gauss-Seidel spectral radii as a function of spacial nodal density. Each table represents a different time interval between solutions of the implicit matric equation. Similarly Tables VI, VII, VIII, and IX list the SOR optimum spectral radii. Figure 11 graphically depicts values from Tables VI, VII, VIII, and IX where x nodal density equals y nodal density. The tables and Figure 11 show that the spectral radius increases with increasing size of time interval. The change in spectral radius with respect to the time interval is largest when the time interval is small and assymptatically approaches zero as the time interval gets large.

These increases in spectral radius cause the iterative matric equation to converge slower. See the discussion on page 22 and Figure 1. When one tries to decrease the truncation error between physical and matric solutions by increasing nodal density, error between the matric and iterative solutions is inadvertently increased by slowing the convergence rate. An example of this is depicted in Figure 12 where the convergence rate using 3 by 3, 4 by 4, and 5 by 5 nodal densities can be compared. Each line represents iterative approximations to the sample physical

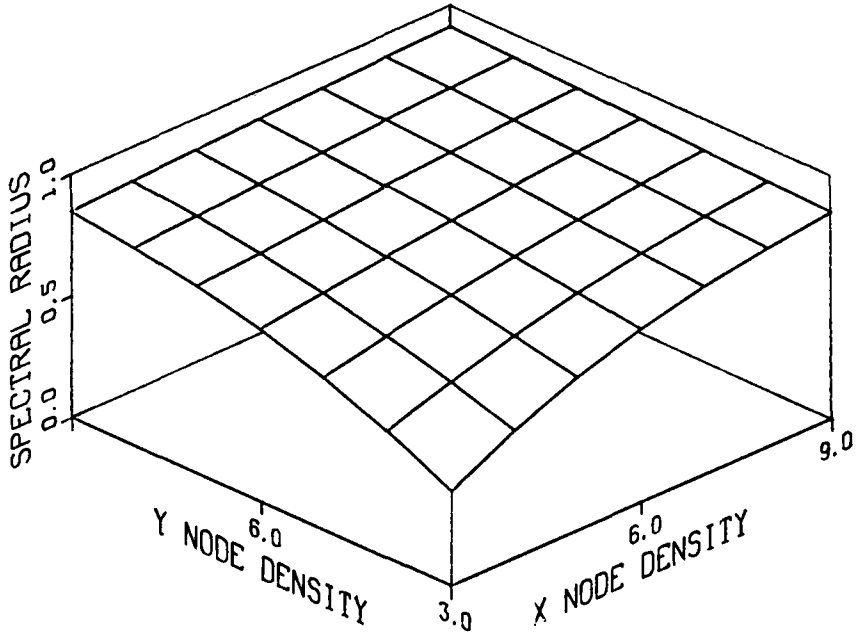


Figure 9. Jacobi Spectral Radius versus Nodal Density, Time Interval = 1

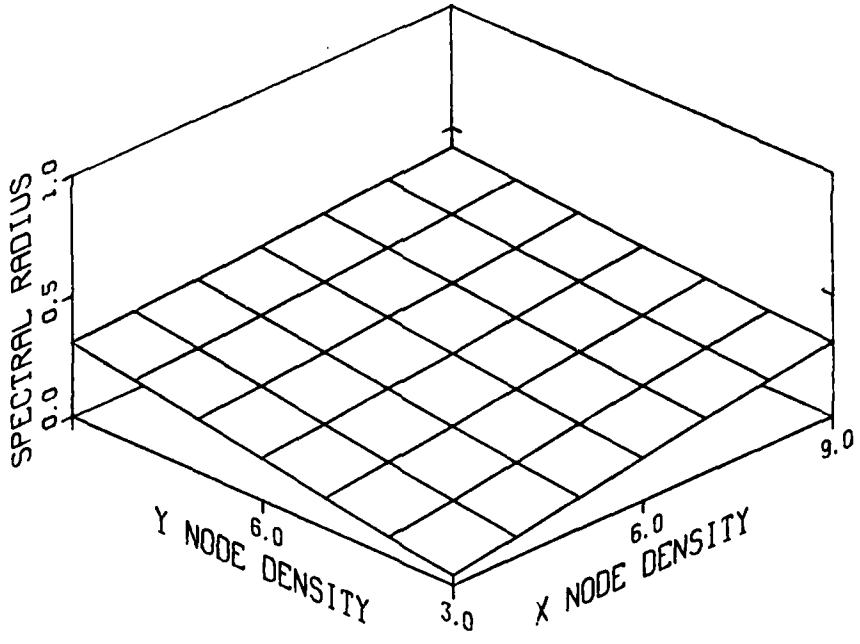


Figure 10. SOR Optimum Spectral Radius versus Nodal Density, Time Interval = 1

Table II

Gauss-Seidel Spectral Radius

Time Interval = 1

**	3*	4*	5*	6*	7*	8*	9*
3	.154	.279	.403	.510	.597	.667	.721
4	.279	.377	.471	.557	.629	.688	.736
5	.403	.471	.542	.608	.665	.714	.754
6	.510	.557	.608	.657	.701	.741	.774
7	.597	.629	.665	.701	.736	.767	.794
8	.667	.688	.714	.741	.767	.791	.814
9	.721	.736	.754	.774	.794	.814	.832

*Number of node intervals along x-axis

**Number of node intervals along y-axis

Table III

Gauss-Seidel Spectral Radius

Time Interval = 10

**	3*	4*	5*	6*	7*	8*	9*
3	.237	.385	.514	.615	.692	.750	.794
4	.385	.487	.578	.656	.717	.766	.804
5	.514	.578	.642	.699	.747	.786	.818
6	.615	.656	.699	.740	.776	.808	.834
7	.692	.717	.747	.776	.804	.828	.849
8	.750	.766	.786	.808	.828	.847	.864
9	.794	.804	.818	.834	.849	.864	.878

*Number of node intervals along x-axis

**Number of node intervals along y-axis

Table IV

Gauss-Seidel Spectral Radius

Time Interval = 100

**	3*	4*	5*	6*	7*	8*	9*
3	.249	.399	.527	.627	.702	.759	.801
4	.399	.500	.590	.667	.727	.775	.812
5	.527	.590	.653	.709	.756	.794	.825
6	.627	.667	.709	.749	.784	.815	.840
7	.702	.727	.756	.784	.811	.835	.855
8	.759	.775	.794	.815	.835	.853	.869
9	.801	.812	.825	.840	.855	.869	.882

*Number of node intervals along x-axis

**Number of node intervals along y-axis

Table V

Gauss-Seidel Spectral Radius

Time Interval = 1000

**	3*	4*	5*	6*	7*	8*	9*
3	.250	.400	.529	.628	.704	.760	.802
4	.400	.502	.592	.668	.728	.775	.812
5	.529	.592	.654	.710	.757	.795	.826
6	.628	.668	.710	.750	.785	.815	.841
7	.704	.728	.757	.785	.812	.835	.856
8	.760	.775	.795	.815	.835	.853	.870
9	.802	.812	.826	.841	.856	.870	.883

*Number of node intervals along x-axis

**Number of node intervals along y-axis

Table VI

SOR Optimum Spectral Radius

Time Interval = 1

**	3*	4*	5*	6*	7*	8*	9*
3	.042	.082	.128	.177	.224	.268	.309
4	.082	.118	.158	.201	.243	.283	.321
5	.128	.158	.193	.230	.267	.303	.337
6	.177	.201	.230	.261	.293	.325	.356
7	.224	.243	.267	.293	.321	.349	.376
8	.268	.283	.303	.325	.349	.373	.397
9	.309	.321	.337	.356	.376	.397	.418

*Number of node intervals along x-axis

**Number of node intervals along y-axis

Table VII

SOR Optimum Spectral Radius

Time Interval = 10

**	3*	4*	5*	6*	7*	8*	9*
3	.067	.121	.178	.234	.286	.333	.375
4	.121	.165	.212	.260	.306	.348	.386
5	.178	.212	.251	.291	.330	.368	.402
6	.234	.260	.291	.325	.358	.390	.421
7	.286	.306	.330	.358	.386	.414	.441
8	.333	.348	.368	.390	.414	.438	.461
9	.375	.386	.402	.421	.441	.461	.482

*Number of node intervals along x-axis

**Number of node intervals along y-axis

Table VIII

SOR Optimum Spectral Radius

Time Interval = 100

**	3*	4*	5*	6*	7*	8*	9*
3	.071	.126	.185	.242	.294	.341	.384
4	.126	.172	.219	.268	.314	.356	.395
5	.185	.219	.259	.299	.338	.376	.410
6	.242	.268	.299	.332	.366	.398	.429
7	.294	.314	.338	.366	.394	.422	.449
8	.341	.356	.376	.398	.422	.446	.469
9	.384	.395	.410	.429	.449	.469	.489

*Number of node intervals along x-axis

**Number of node intervals along y-axis

Table IX

SOR Optimum Spectral Radius

Time Interval = 1000

**	3*	4*	5*	6*	7*	8*	9*
3	.072	.127	.186	.243	.295	.342	.384
4	.127	.172	.220	.269	.314	.357	.395
5	.186	.220	.260	.300	.339	.377	.411
6	.243	.269	.300	.333	.367	.399	.430
7	.295	.314	.339	.367	.395	.423	.449
8	.342	.357	.377	.399	.423	.446	.470
9	.384	.395	.411	.430	.449	.470	.490

*Number of node intervals along x-axis

**Number of node intervals along y-axis

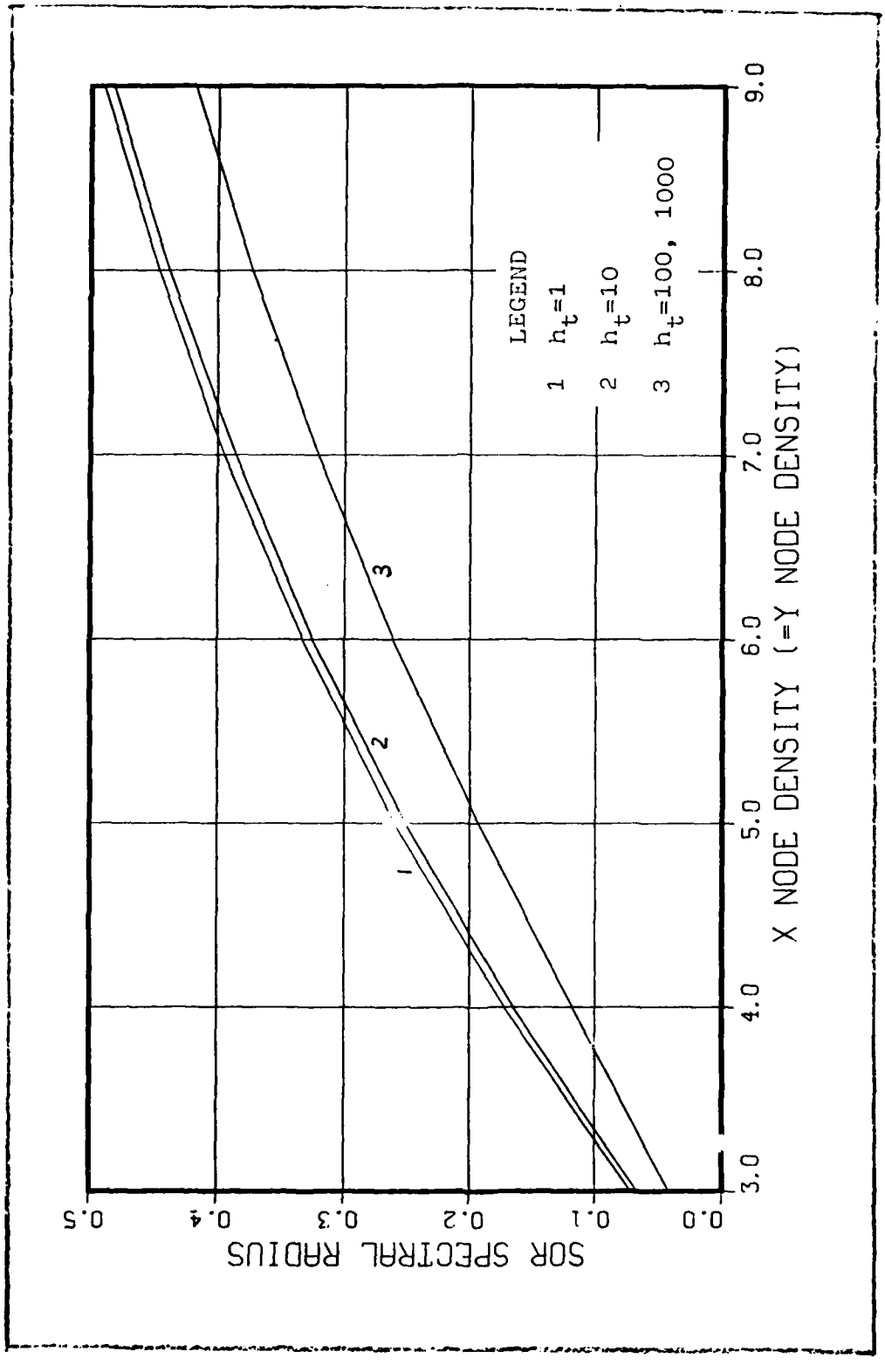


Figure 11. Optimum SOR Spectral Radius versus Time Step Interval

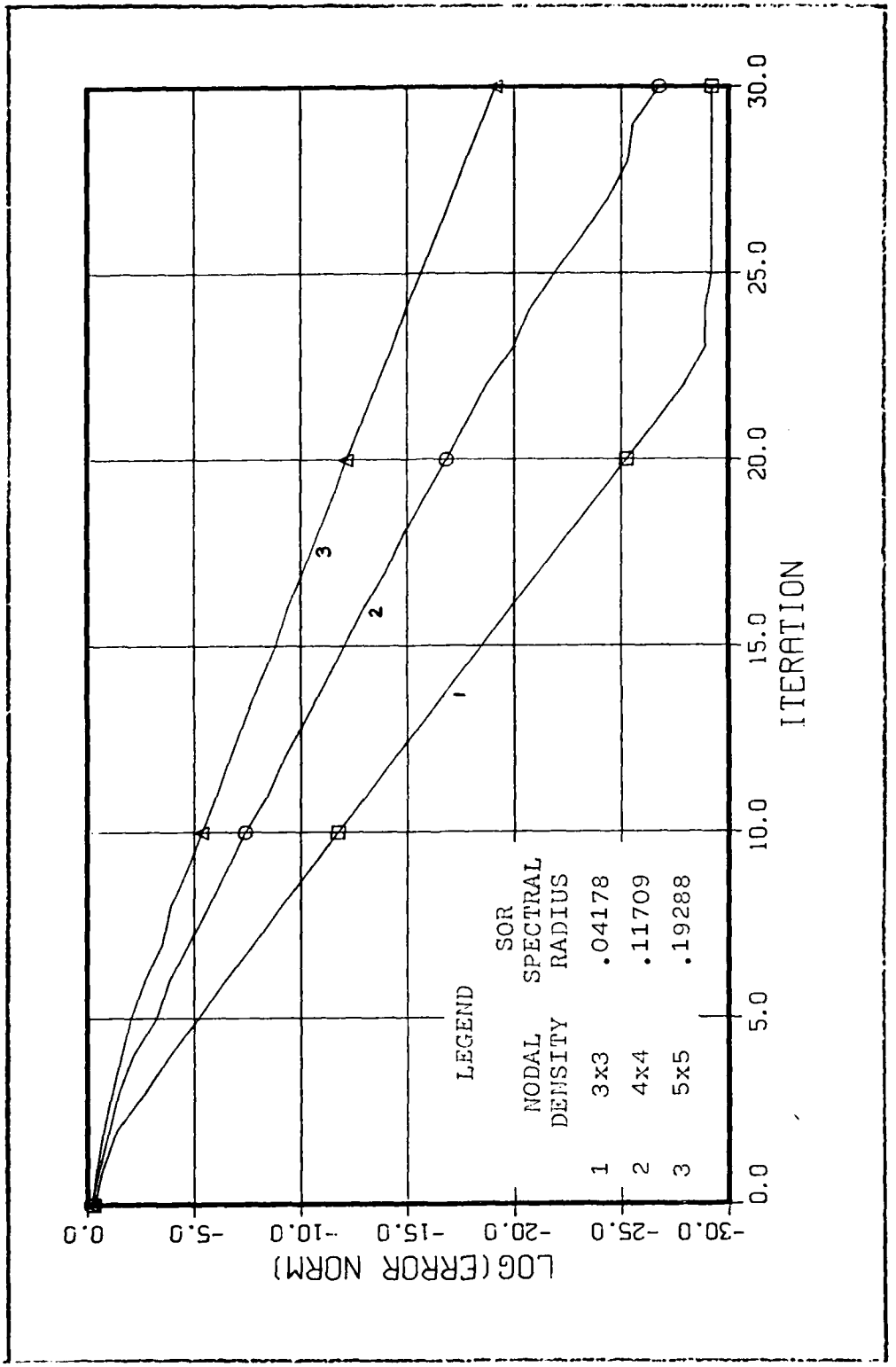


Figure 12. Convergence Rate versus Nodal Density

problem described earlier.

Iterative Error Approximations

The computed error data is displayed in sequences of nine graphs each. Each sequence represents results using a different nodal density over the same spacial domain of the sample problem. Graphs representing a domain with ten nodal intervals on a side are included in this section. The data was determined by using the following selected parameters and resulting constants.

1. Number of interior nodes = 81
2. Time interval between matric solutions = 10,000
3. Diffusion coefficient = 1
4. Computer precision = 27 decimal digits
5. Number of spectral radius iterations = 60
6. Jacobi spectral radius = .95105
7. Gauss-Seidel spectral radius = .90451
8. SOR optimum spectral radius = .52786
9. Number of iterations = 200

Graphs representing other nodal densities are included as Appendices H and I.

The first three graphs of a sequence depict relationships between error limit parameters, displacement norms, and error norms when using

$$\|e^{(n)}\| \leq \|d^{(n)}\| / (1-K) \tag{6}$$

as the error limit method. The second three graphs depict relationships when using

$$\|e^{(n)}\| \leq \|d^{(n-1)}\| K / (1-K) \tag{7}$$

as the error limit method. The seventh graph depicts relationships when using

$$\|e^{(n)}\| \leq \|d^{(0)}\| K^n / (1-K) \tag{8}$$

as the error limit method. Henceforth these methods will be labeled one, two, and three respectively. Note, here all norms are infinite-norms.

The first graph, Figure 13 for example, shows the error norm to displacement norm ratio related to the first error limit method. The second graph, Figure 14 for example, shows the parameter K needed to make Equation (6) an equality. Candidate parameters may be compared to this minimum parameter. The third graph, Figure 15 for example, depicts the displacement norm and a set of parallel curves Each parallel curve represents

$$\|\delta^{(n)}\| /(1-K)$$

where the parameter K is chosen to space the curves an order of magnitude apart. The error norm is superimposed so that an order of magnitude comparison can be made between the error norm, the displacement norm and candidate error limits based on different parameters.

Similarly, the fourth graph, Figure 16 for example, shows the error norm to displacement norm ratio related to the second error limit method. The fifth graph, Figure 17 for example, shows the parameter K needed to make Equation (7) an equality. Candidate parameters again may be compared to this minimum parameter. The sixth graph, Figure 18 for example, depicts the displacement norm and a set of parallel curves. Each parallel curve represents

$$\|\delta^{(n-1)}\| K/(1-K)$$

where the parameter K is chosen to space the curves an order of magnitude apart. Here again, the error norm is superimposed so that an order of magnitude comparison can be made between the error norm, the displacement norm, and candidate error limits based on different parameters.

The seventh graph, Figure 19 for example, pertains to the third error limit method. The graph depicts straight lines representing

$$\| d^{(0)} \| \, K^n / (1-K)$$

where the parameter K is incremented by one tenth from zero to unity.

(20)
$$\log \left[\| d^{(0)} \| \, K^n / (1-K) \right]$$
$$= n \log(K) + \log \| d^{(0)} \| - \log (1-K)$$

This is in the form of a straight line

$$y = m x + b$$

where

$$y = \log \left[\| d^{(0)} \| \, K^n / (1-K) \right]$$
$$x = n$$
$$b = \log \| d^{(0)} \| - \log (1-K)$$
$$m = \log (K)$$

The error norm is superimposed so that an order of magnitude comparison can be made between the error norm and candidate error limits based on different parameters K.

The eighth graph, Figure 20 for example, depicts the displacement norm ratio. It is a candidate error limit parameter, but unlike the other candidate parameters, it is not constant. The displacement norm ratio can also be compared to its limiting value, the spectral radius.

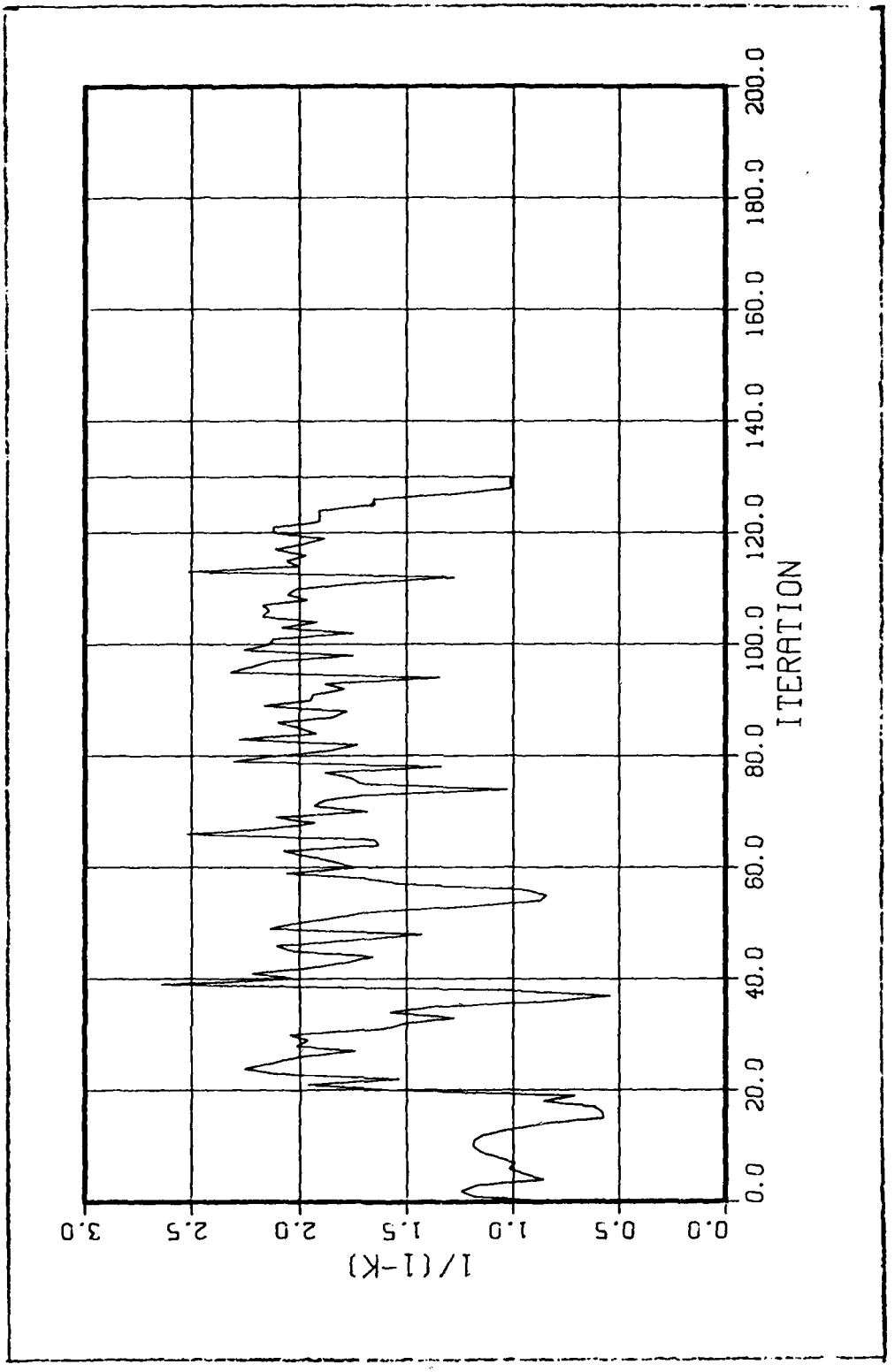


Figure 13. Error Norm to Displacement Norm Ratio, Error Limit Method 1, 10 x 10 Modal Density

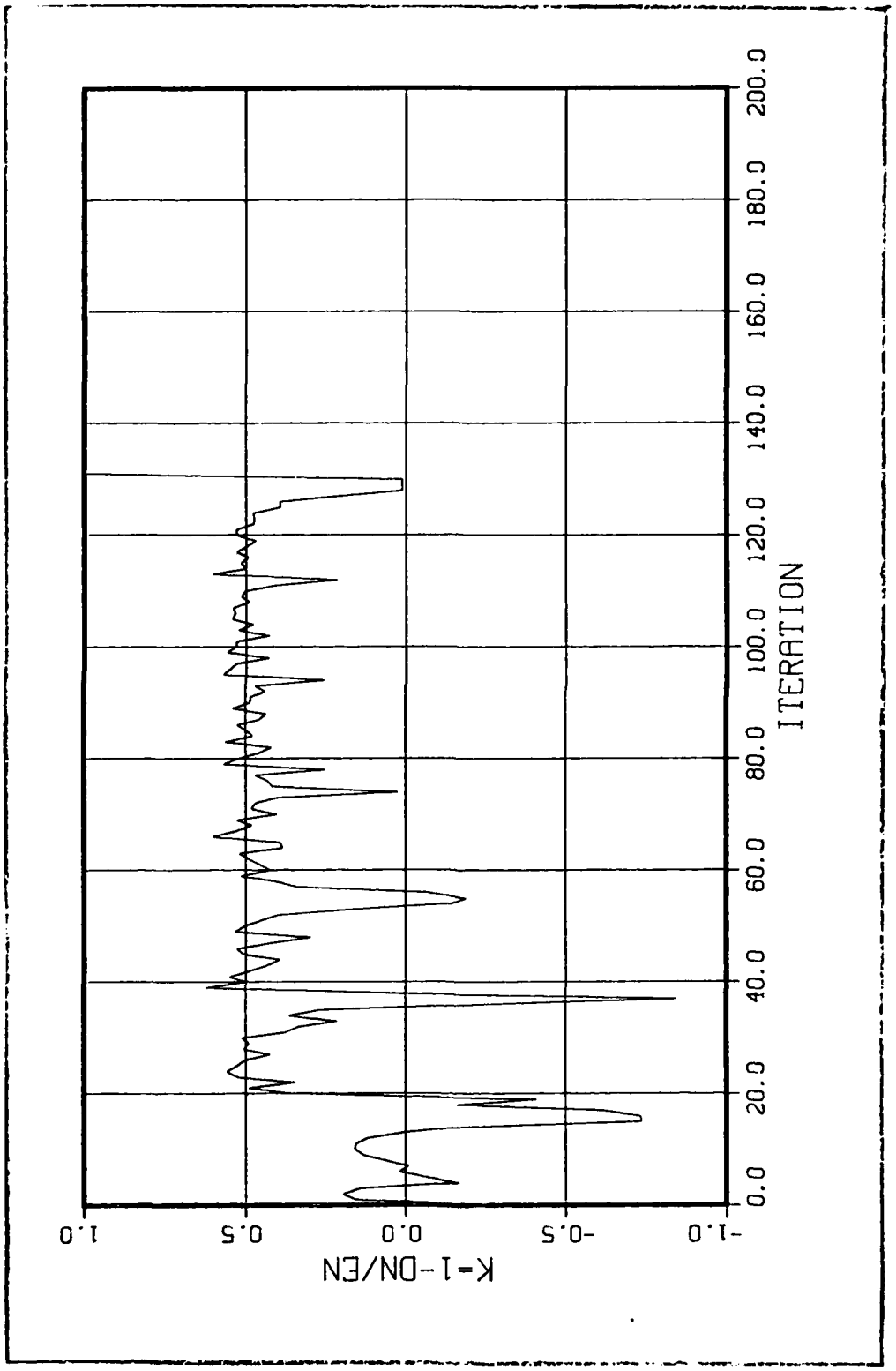


Figure 14. Parameter Required for Error Limit Method 1 to Equal the Error Norm, 10 x 10 Nodal Density

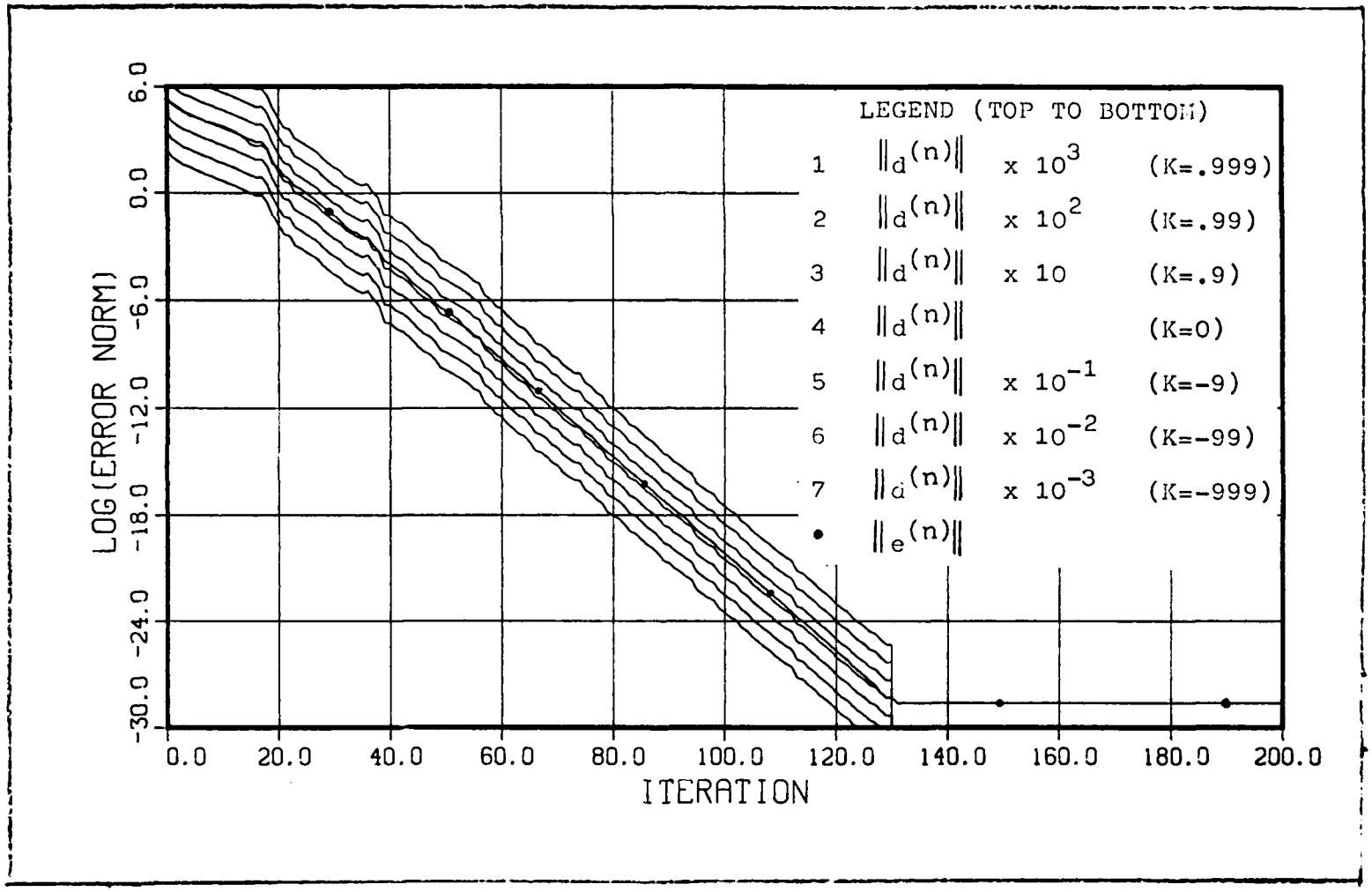


Figure 15. Order of Magnitude Comparison between Error Norm and Displacement Norm, Error Limit Method 1, 10 x 10 Nodal Density

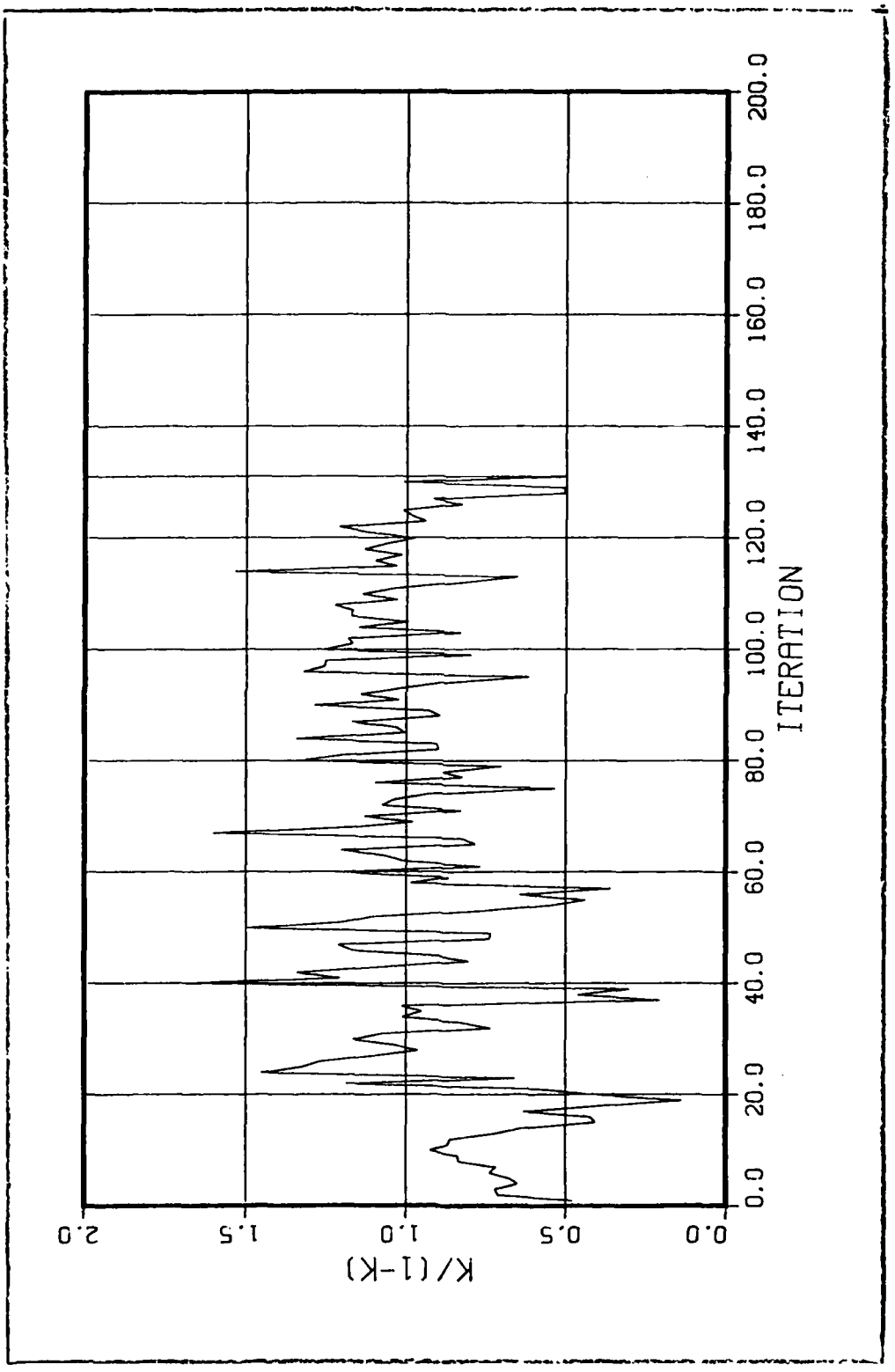


Figure 16. Error Norm to Displacement Norm Ratio, Error Limit Method 2, 10 x 10 Nodal Density

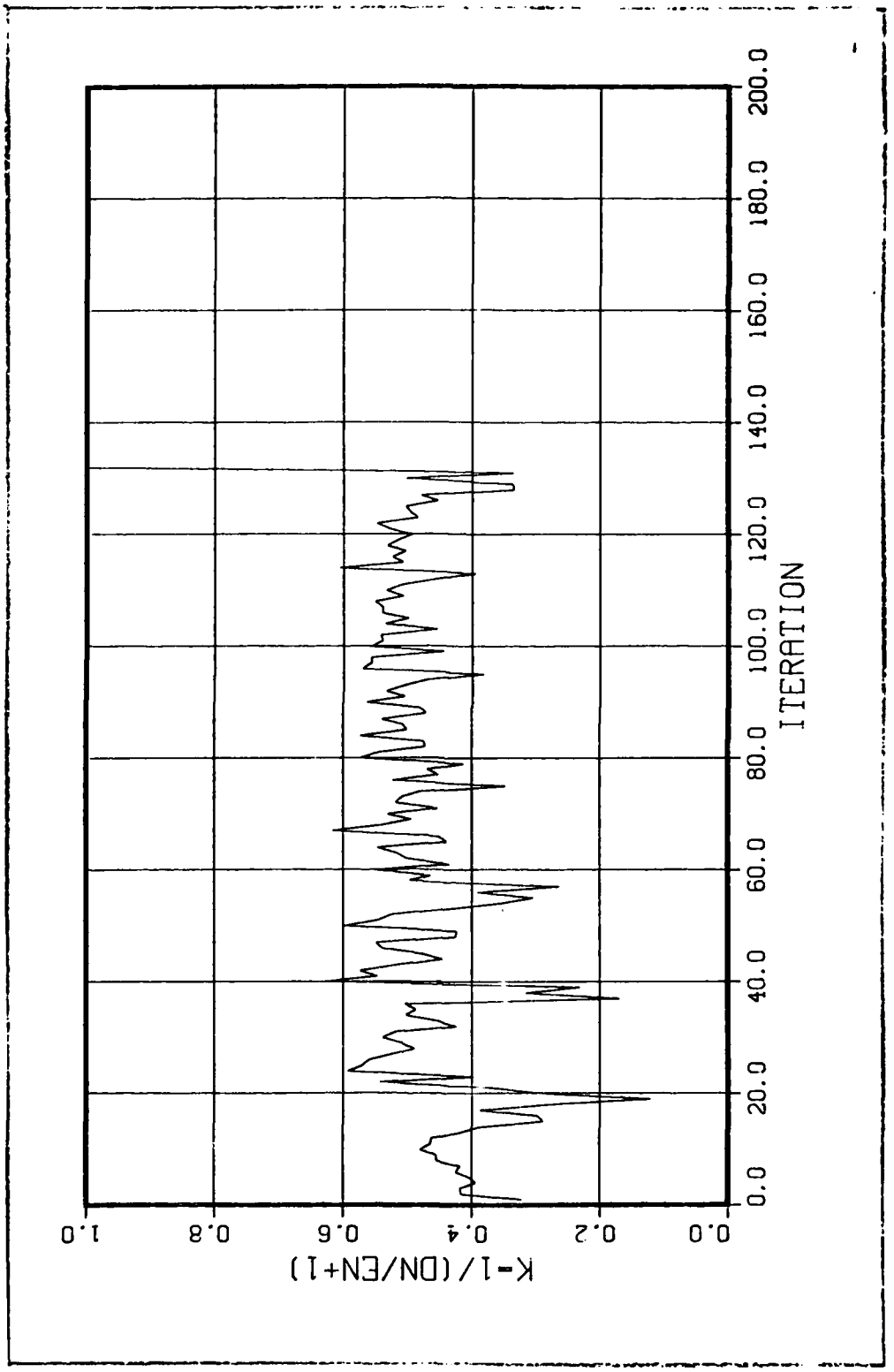


Figure 17. Parameter Required for Error Limit Method 2 to Equal the Error Norm, 10 x 10 Nodal Density

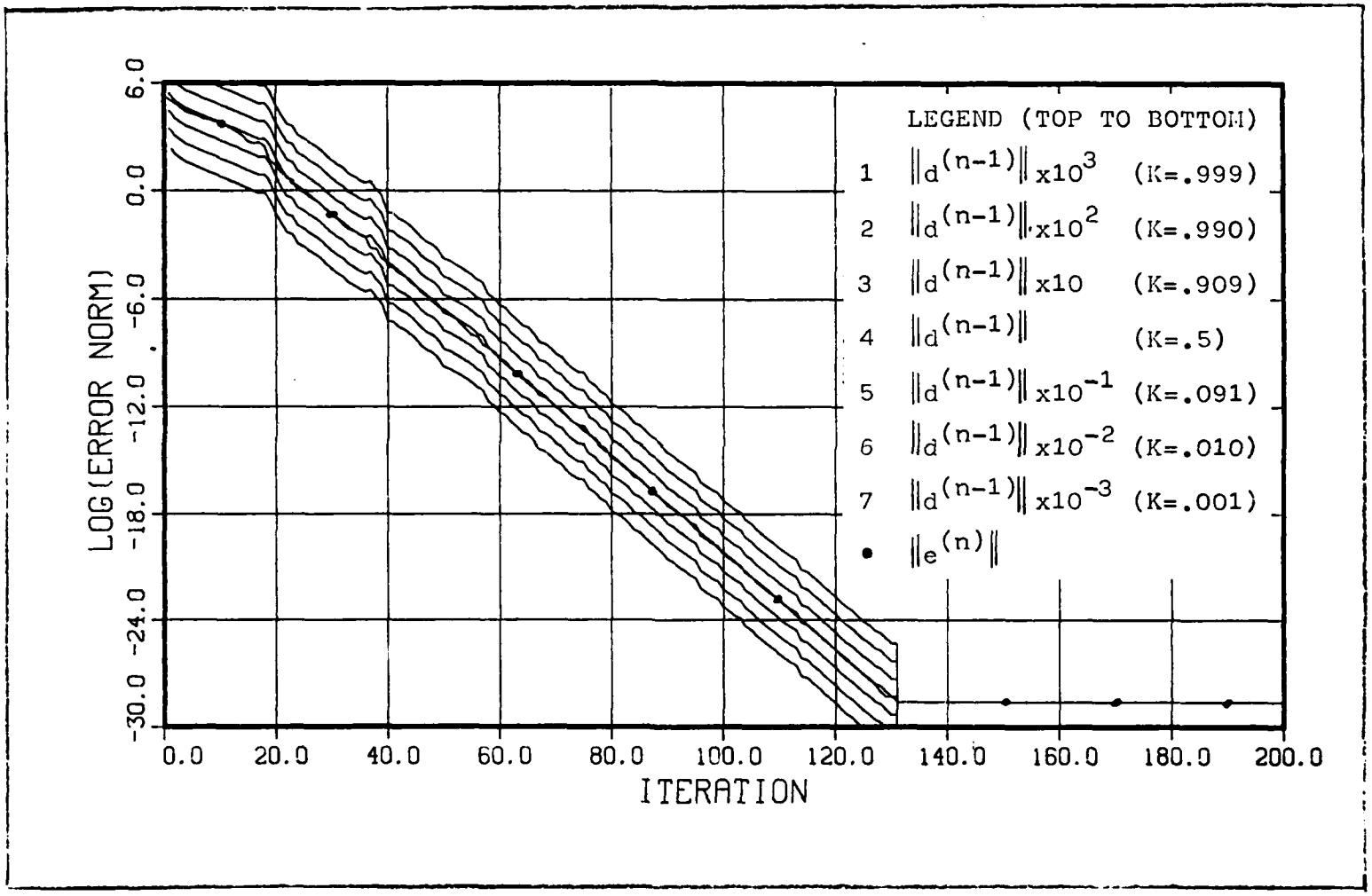


Figure 18. Order of Magnitude Comparison between Error Norm and Displacement Norm, Error Limit Method 2, 10 x 10 Nodal Density

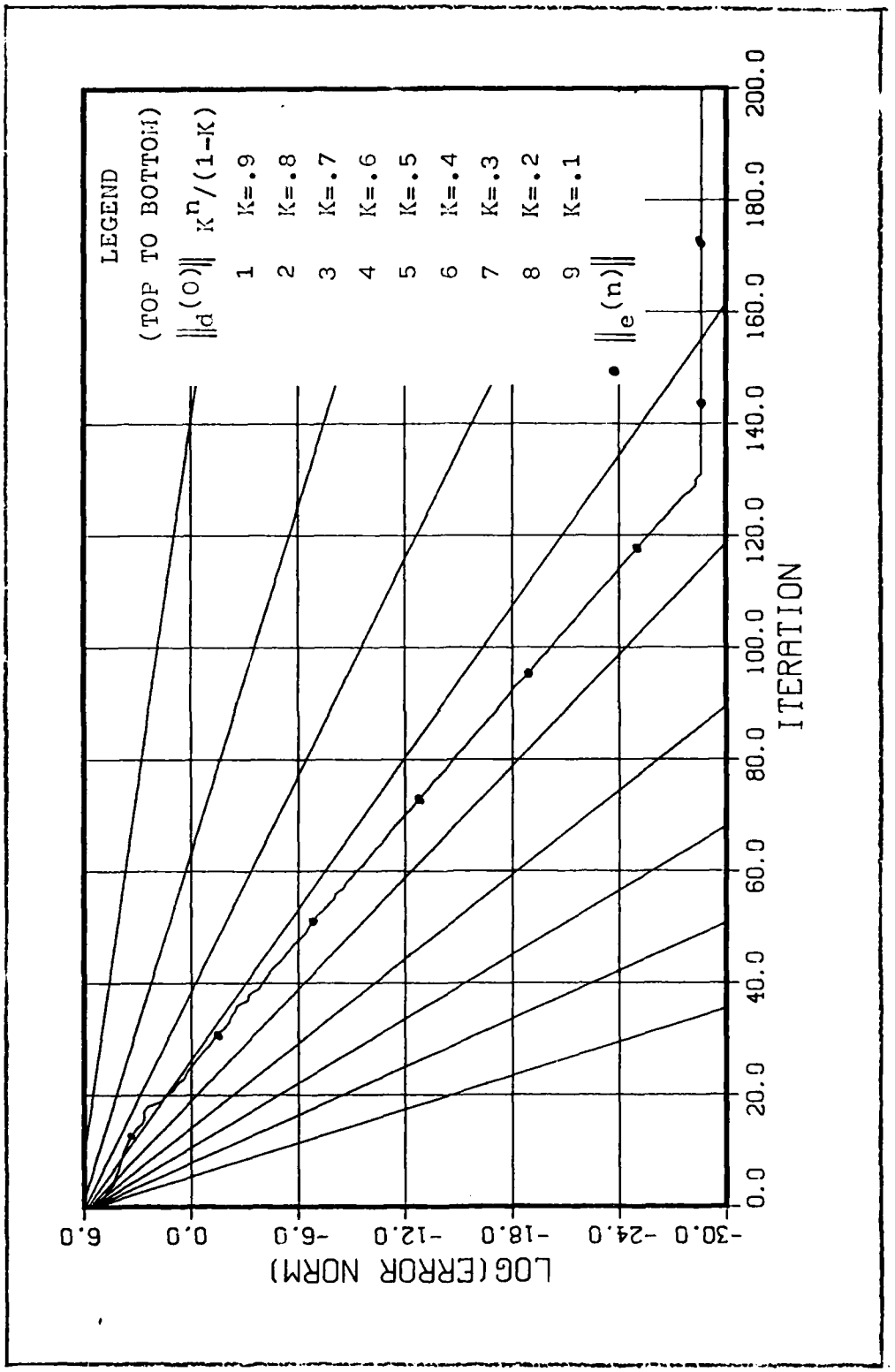


Figure 19. Comparison of Error Norm to Error Limit Method 3, 10 x 10 Nodal Density

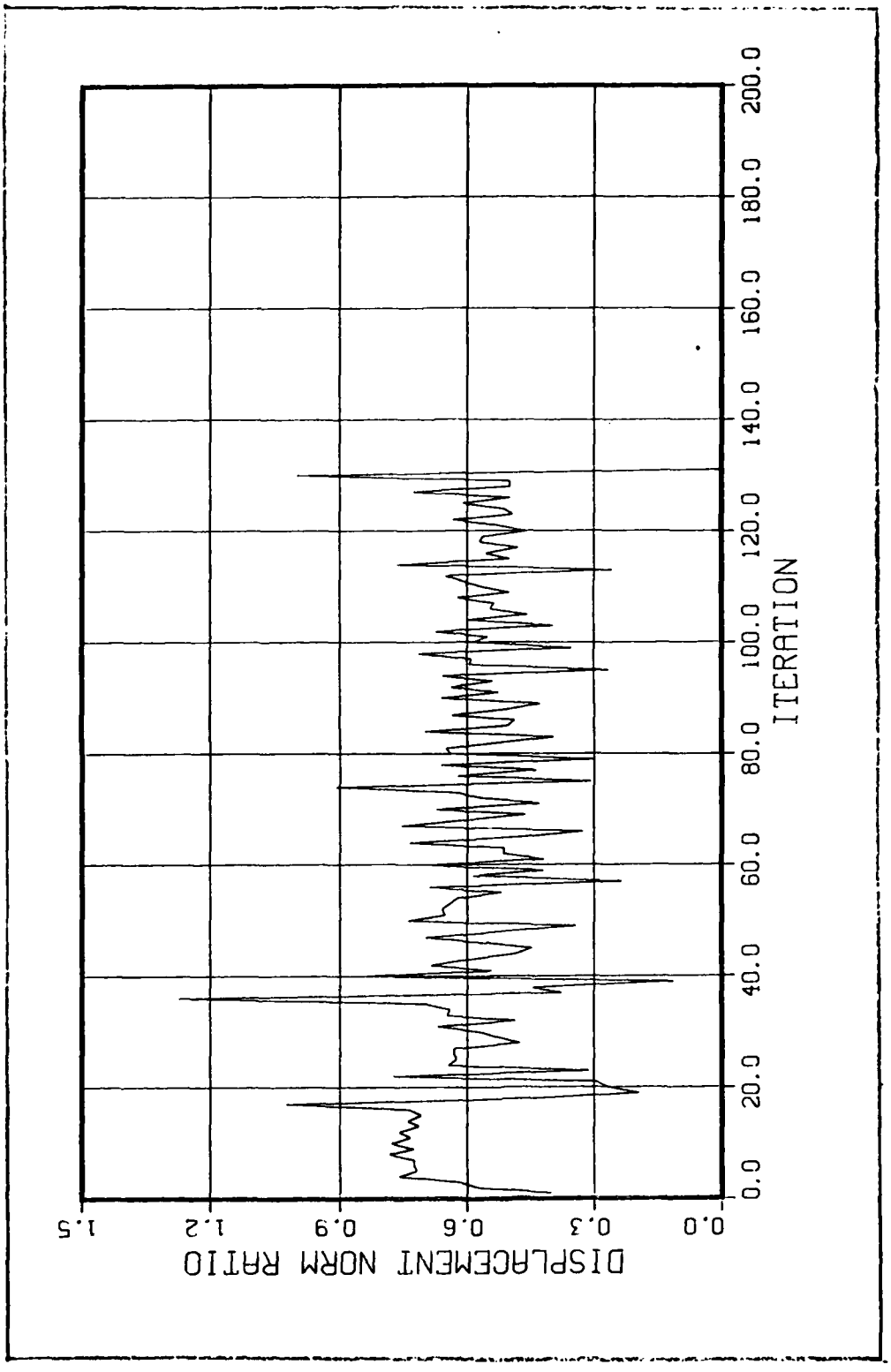


Figure 20. Displacement Norm Ratio, 10 x 10 Modal Density

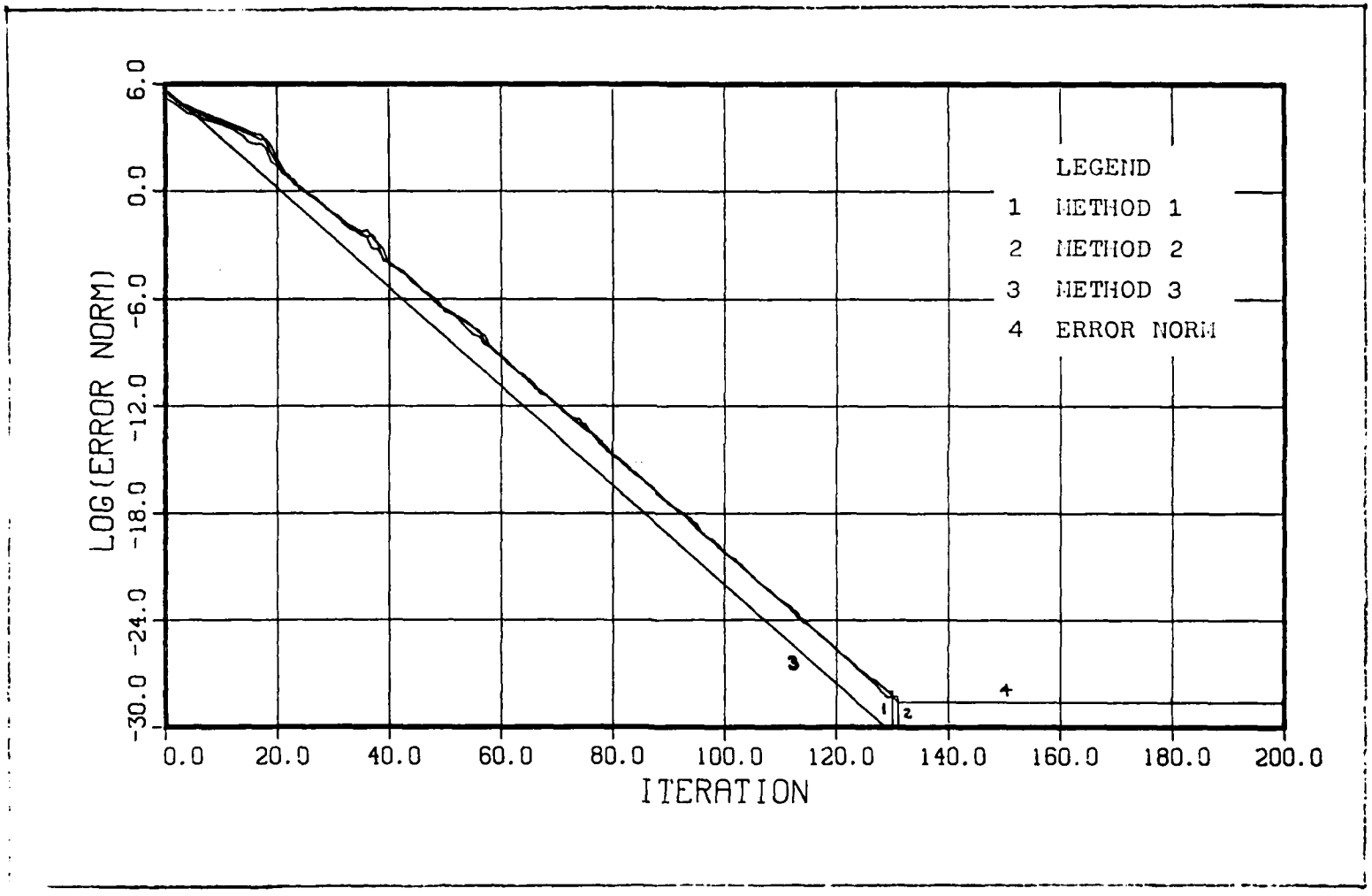


Figure 21. Three Error Limit Methods Compared to Error Norm, K = SOR Optimum Spectral Radius, 10 x 10 Nodal Density

The ninth graph, Figure 21 for example, compares the error norm to the three error limit methods. The parameter K in all error limit methods being the same.

In comparing error limit method one to error limit method two, it can easily be shown that when the displacement norm ratio (Figure 20) is less than the parameter K

$$\|d^{(n)}\| / \|d^{(n-1)}\| < K$$

then method one is less than method two

$$\|d^{(n)}\| / (1-K) < \|d^{(n-1)}\| K/(1-K)$$

while when

$$\|d^{(n)}\| / \|d^{(n-1)}\| > K$$

method two is less than method one

$$\|d^{(n-1)}\| K/(1-K) < \|d^{(n)}\| / (1-K)$$

When the ratio equals the parameter,

$$\|d^{(n)}\| / \|d^{(n-1)}\| = K$$

both methods give the same result

$$\|d^{(n-1)}\| K/(1-K) = \|d^{(n)}\| / (1-K)$$

This does not imply which method approximated the error more closely. That depends on which candidate parameter is used in each method. Figures 14 and 17 show the parameters that would make each approximation an equality. The following candidate parameters for the 10

by 10 nodal density can be compared with the graphs in this section.

1. Displacement norm ratio: Figure 18
2. Jacobi spectral radius = .951
3. Gauss-Seidel spectral radius = .905
4. SOR optimum spectral radius = .528
5. Zero

Note that on all the graphs representing 10 by 10 nodal density, computer truncation error overrides any further iterative convergence past iteration number 130.

Figures 14 and 17 show that the optimum error limit parameter oscillates about a value close to the SOR optimum spectral radius. The fact that the SOR optimum spectral radius is the limiting value of the optimum K, makes it a good choice. The Jacobi and Gauss-Seidel spectral radii, in this case, fall above all optimum values of K. Using a parameter equal to zero in method one or equal to one half in method two essentially approximates the error by the displacement. It is neither a lower limit, upper limit, nor as good an approximation as when using the SOR optimum spectral radius.

Figures 15 and 18 show order of magnitude comparisons between the error norm and approximations using different parameters. Note in particular that as K approaches unity, the order of magnitude changes rapidly. These two error limit methods are very sensitive to changes in parameter when the parameter approaches unity, in the limiting case, when the spectral radius approaches unity.

Note that the error norm roughly parallels the displacement norm, as it should in the limit as the number of iterations increase. Both lines have a slope approaching the logarithm of the spectral radius. Similar to Equations (11) and (12)

$$\|e^{(n+p)}\| = |\lambda|^n \|e^{(n)}\|$$

$$\log \|e^{(n+p)}\| = n \log|\lambda| + \log \|e^{(p)}\|$$

where the slope is

$$m = \log|\lambda|$$

and

$$\|d^{(n+p)}\| = |\lambda|^n \|d^{(p)}\|$$

$$\log \|d^{(n+p)}\| = n \log|\lambda| + \log \|d^{(p)}\|$$

where again the slope is

$$m = \log|\lambda|$$

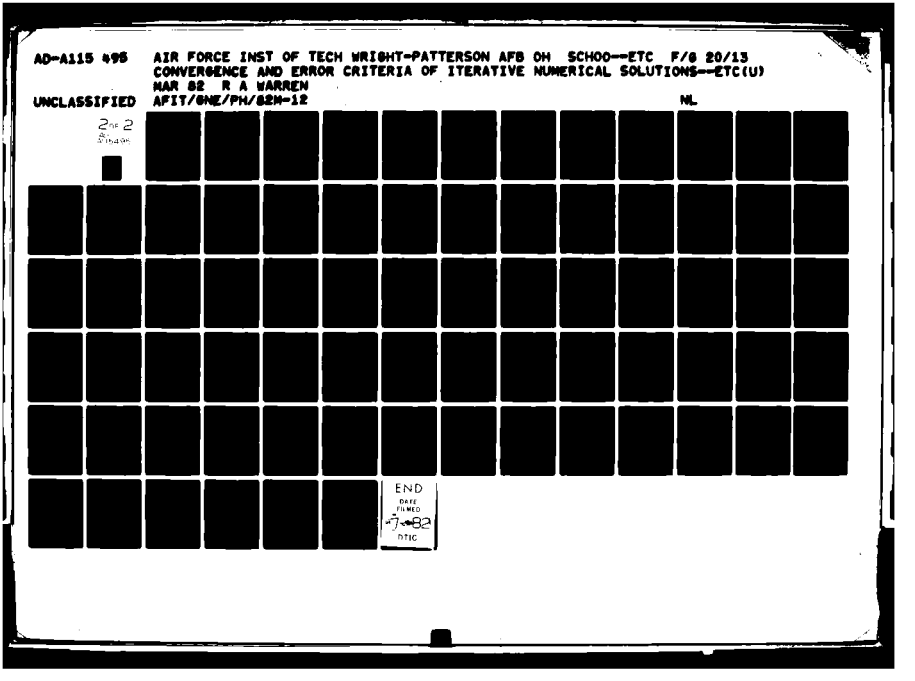
Figure 19 shows a comparison between the error norm and limit method three. Note that the approximation will diverge from the error norm unless both have the same slope. From Equation (20), the slope of the approximation is

$$m = \log(K)$$

therefore the parameter must equal the spectral radius for the approximation to parallel the error norm in the limit as iteration increases.

Figure 21 shows a comparison of the three error limit methods and the error norm. The parameter was chosen to equal the SOR spectral radius. Note that method three does not approximate the error norm as well as methods one and two. This was typical. See Figures 30 and 39.

Figure 20 shows that the displacement norm ratio appears to oscillate about a value close to the SOR spectral radius. From Equation (12), the limiting value of the


AD-A115 498
AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 20/13
CONVERGENCE AND ERROR CRITERIA OF ITERATIVE NUMERICAL SOLUTIONS--ETC(U)
MAR 82 R A WARREN
UNCLASSIFIED AFIT/GNE/PH/82M-12 NL
2 OF 2
END
DATE FILMED
7-82
DTIC

displacement norm ratio is in fact equal to the spectral radius. Averaging the displacement ratio over a number of previous iterations is one way to approximate the spectral radius.

One advantage of method two over method one might be the fact that the domain of the parameter K for method two (Figure 5) is the same as the range of possible convergent spectral radius values, namely, zero to unity.

With respect to additional computer operations, methods two and three require only the limit method equation be algebraically solved. Method one requires an additional iteration step to determine the nth displacement.

$$x^{(n)} = x^{(n-1)} + d^{(n-1)}$$

$$d^{(n)} = G\, d^{(n-1)}$$

$$\|e^{(n)}\| \simeq \|d^{(n)}\| / (1-K)$$

where as the displacement needed for method two is already known

$$x^{(n)} = x^{(n-1)} + d^{(n-1)}$$

$$\|e^{(n)}\| \simeq \|d^{(n-1)}\|\, K/(1-K)$$

and method three uses only the initial displacement.

$$x^{(n)} = x^{(n-1)} + d^{(n-1)}$$

$$\|e^{(n)}\| \simeq \|d^{(0)}\|\, K^n/(1-K)$$

Method two appears to be the best choice with respect to required computer operations, accuracy, and the intrinsic domain of its parameter being zero to unity.

The common method of testing for convergence is to use the displacement as an approximation of the error between matric and iterative solutions. The displacement will be within an order of magnitude of the error as long as the actual parameter K associated with method two (Figure 17) remains between 0.1 and 0.9 (Figure 18). Remembering that K approaches the spectral radius as the number of iterations become large suggests that the displacement can be used when the spectral radius is between 0.1 and 0.9.

Errors can be expressed in absolute or relative terms. One is not intrinsically more accurate than the other. The choice is a matter of convenience in expression. A method which yields a more accurate absolute error approximation will yield a more accurate relative error approximation. So this discussion is applicable to both expressions.

V Conclusions and Recommendations

Conclusions

1. Of those methods investigated, and depending on the parameter K chosen,

$$\|e^{(n)}\| \simeq \|d^{(n-1)}\| \; K/(1-K)$$

represents the optimum error approximation method with respect to accuracy, ease of computing, and required computer resources.

2. The optimum choice for parameter K is the spectral radius of the iterative matrix. It can be calculated using the power method and equations relating Jacobi spectral radius to SOR spectral radius. Alternately the spectral radius can be approximated by averaging displacement norm ratios of previous iterations.

3. The displacement norm will normally approximate the error norm within an order of magnitude when the spectral radius is between 0.1 and 0.9.

4. The spectral radius of the iterative matrix increases as spacial nodal density increases, thus convergence rate decreases.

5. The spectral radius of the iterative matrix increases as the time interval between solutions of the implicit matric equation is increased. Thus convergence rate decreases.

Recommendations

Since the Jacobi iteration matrix for transient heat conduction is simple and retains its simple form as spacial nodal density is varied, a particularly simple similarity transformation may exist that diagonalizes the Jacobi matrix. The Jacobi eigenvalues and spectral radius could then be obtained from it rather than from a lengthy iterative procedure such as the power method.

Bibliography

1. Arfken, George. Mathematical Methods for Physicists. New York: Academic Press, Inc., 1968.
2. Carslaw, Horatio Scott and John Conrad Jaeger. Conduction of Heat in Solids (Second Edition). London: Oxford University Press, 1959.
3. Chemical Rubber Company. Standard Mathematical Tables (Seventeenth Edition), edited by Samuel M. Selby. Cleveland: The Chemical Rubber Co., 1969.
4. Churchill, Ruel Vance and James Ward Brown. Fourier Series and Boundary Value Problems (Third Edition). New York: McGraw-Hill Book Company, 1978.
5. Clark, Melville Jr. and Kent F. Hansen. Numerical Methods of Reactor Analysis. New York: Academic Press Inc., 1964.
6. Conte, Samuel Daniel and Carl de Boor. Elementary Numerical Analysis (Third Edition). New York: McGraw-Hill Book Company, 1980.
7. Faddeev, D.K. and V.N. Faddeeva. Computational Methods of Linear Algebra, translated by Robert C. Williams, edited by R.A. Rosenbaum and G. Philip Johnson. San Francisco: W.H. Freeman and Company, 1963.
8. Hageman, L.A. Adaptive Procedures for the SOR Method. Unpublished PhD dissertation.
9. Kaplan, Bernard, Professor of Physics. Lecture materials distributed in PH 6.85. Methods of Engineering Analysis. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1981.
10. Kreyszig, Erwin. Advanced Engineering Mathematics (Third Edition). New York: John Wiley and Sons, Inc., 1972.
11. Moise, Edwin E. Calculus. Reading, Mass.: Addison-Wesley Publishing Company, 1967.
12. Smith, Gordon G. Numerical Solution of Partial Differential Equations (Second Edition). Oxford: Clarendon Press, 1978.

Appendix A

Eigenvalue Error Limit

Derivation of the error limit between the quotient approximation and the eigenvalue closest to it (10:462-463).

e_i = orthonormal eigenvectors of matrix A.

$x = (c_i e_i)$, expanded in terms of eigenvectors.

$$y = A x$$

$$= A \sum_{i=1}^{n} (c_i e_i)$$

$$= \sum_{i=1}^{n} (c_i A e_i)$$

$$A e_i = \lambda e_i$$

$$= \sum_{i=1}^{n} (c_i \lambda_i e_i)$$

v = arbitrary vector

$$q_1 \equiv v^T y / (v^T x)$$

$$y - q_1 x = \sum_{i=1}^{n} (c_i \lambda_i e_i) - q_1 \sum_{i=1}^{n} (c_i e_i)$$

$$= \sum_{i=1}^{n} (c_i [\lambda_i - q_1] e_i)$$

$$(y - q_1 x)^T (y - q_1 x) = \sum_{i=1}^{n} (c_i^2 [\lambda_i - q_1]^2 e_i^2)$$

$\lambda_i - q_1$ = real

$$[\lambda_i - q_1]^2 = |\lambda_i - q_1|^2$$

$$(y-q_1x)^T(y-q_1x) = \sum_{i=1}^{n} (c_i^2 |\lambda_i - q_1|^2 e_i^2)$$

$\lambda_q = \lambda_i$ closest to q_i

$|\lambda_q - q_1| \leq |\lambda_i - q_1|$ for all i

$$\geq \sum_{i=1}^{n} (c_i^2 |\lambda_q - q_1|^2 e_i^2)$$

$$= |\lambda_q - q_1|^2 \sum_{i=1}^{n} (c_i^2 e_i^2)$$

$\sum_{i=1}^{n} (c_i^2 e_i^2) = x^T x$

$\equiv m_0$

$$= |\lambda_q - q_1|^2 m_0$$

$|\lambda_q - q_1| \equiv \epsilon_1$

$$= \epsilon_1^2 m_0$$

$(y-q_1x)^T = y^T - (q_1x)^T$

q_1 = scalar

$$= y^T - q_1 x^T$$

$$(y-q_1x)^T(y-q_1x) = (y^T - q_1x^T)(y-q_1x)$$

$$= y^Ty - y^Tq_1x - q_1x^Ty + q_1^2x^Tx$$

$$= q_1^2(x^Tx) - q_1(y^Tx + x^Ty) + (y^Ty)$$

y^Tx = symmetric, scalar

$y^Tx = (y^Tx)^T$

$$y^T x = x^T y \qquad (1:162,165)$$

$$(y - q_1 x)^T (y - q_1 x) = q_1^2 (x^T x) - 2 q_1 (x^T y) + (y^T y)$$

$$m_0 \equiv x^T x$$

$$m_1 \equiv x^T y$$

$$m_2 \equiv y^T y$$

$$= q_1^2 m_0 - 2 q_1 m_1 + m_2$$

$$= m_0 (q_1^2 - 2 q_1 m_1 / m_0 + m_2 / m_0)$$

$$\delta_1 \equiv \sqrt{(q_1^2 - 2 q_1 m_1 / m_0 + m_2 / m_0)}$$

$$= m_0 \delta_1^2$$

$$\epsilon_1^2 m_0 \leq (y - q_1 x)^T (y - q_1 x) = m_0 \delta_1^2$$

$$m_0 > 0$$

$$\epsilon_1^2 \leq \delta_1^2$$

$$\epsilon_1 \leq \delta_1$$

$$= \sqrt{(q_1^2 - 2 q_1 m_1 / m_0 + m_2 / m_0)}$$

Similarly

e_i = orthonormal eigenvectors of matrix A.

$X = \sum_{i=1}^{n} (c_i e_i)$, expanded in terms of eigenvectors.

$$y = Ax$$

$$z = Ay$$

$$= A^2 x$$

$$= A^2 \sum_{i=1}^{n} (c_i e_i)$$

$$= \sum_{i=1}^{n} (c_i A^2 e_i)$$

$$Ae_i = \lambda_i e_i$$

$$= \sum_{i=1}^{n} (c_i \lambda_i^2 e_i)$$

v = arbitrary vector

$$q_2 \equiv v^T z / (v^T x)$$

$$z - q_2 x = \sum_{i=1}^{n} (c_i \lambda_i^2 e_i) - q_2 \sum_{i=1}^{n} (c_i e_i)$$

$$= \sum_{i=1}^{n} (c_i [\lambda_i^2 - q_2] e_i)$$

$$(z - q_1 x)^T (z - q_1 x) = \sum_{i=1}^{n} (c_i^2 [\lambda_i^2 - q_2]^2 e_i^2)$$

$\lambda_i^2 - q_2$ = real

$$[\lambda_i^2 - q_2]^2 = |\lambda_i^2 - q_2|^2$$

$$= \sum_{i=1}^{n} (c_i^2 |\lambda_i^2 - q_2|^2 e_i^2)$$

$\lambda_q = \lambda_i$ closest to

$|\lambda_q^2 - q_2| \leq |\lambda_i^2 - q_2|$ for all i

$$\geq \sum_{i=1}^{n} (c_i^2 |\lambda_q^2 - q_2|^2 e_i^2)$$

$$= |\lambda_q^2 - q_2|^2 \sum_{i=1}^{n} (c_i^2 e_i^2)$$

$$\sum_{i=1}^{n} (c_i^2 e_i^2) = x^T x$$

$$\equiv m_0$$

$$= |\lambda_q^2 - q_2|^2 m_0$$

$$|\lambda_q^2 - q_2| \equiv \epsilon_2$$

$$= \epsilon_2^2 m_0$$

$$(z - q_2 x)^T = z^T - (q_2 x)^T$$

q_2 = scalar

$$= z^T - q_2 x^T$$

$$(z - q_2 x)^T (z - q_2 x) = (z^T - q_2 x^T)(z - q_2 x)$$

$$= z^T z - z^T q_2 x - q_2 x^T z + q_2^2 x^T x$$

$$= q_2^2 (x^T x) - q_2 (z^T x + x^T z) + z^T z$$

$z^T x$ = symmetric, scalar

$$z^T x = (z^T x)^T$$

$$= x^T z \qquad (1:162,165)$$

$$= q_2^2 (x^T x) - 2 q_2 (x^T z) + (z^T z)$$

$$m_0 \equiv x^T x$$

$$m_3 \equiv x^T z$$

$$m_4 \equiv z^T z$$

$$= q_2^2 m_0 - 2 q_2 m_3 + m_4$$

$$= m_0 (q_2^2 - 2 q_2 m_3 / m_0 + m_4 / m_0)$$

$$\delta_2 \equiv \sqrt{(q_2^2 - 2 q_2 m_3 / m_0 + m_4 / m_0)}$$

$$= m_0 \delta_2^2$$

$$\epsilon_2^2 m_0 \leq (z - q_2 x)^T (z - q_2 x) = m_0 \delta_2^2$$

$$m_0 > 0$$

$$\epsilon_2^2 \leq \delta_2^2$$

$$\epsilon_2 \leq \delta_2$$

$$= \sqrt{(q_2^2 - 2 q_2 m_3 / m_0 + m_4 / m_0)}$$

Must now find a limit on $\epsilon_3 = |\lambda_q - \sqrt{(q_2)}|$ from δ_2

$$\delta_2 \geq \epsilon_2 = |\lambda_q^2 - q_2| \geq 0$$

If $\lambda_q^2 \geq q_2 \geq 0$

then $|\lambda_q^2 - q_2| = \lambda_q^2 - q_2 \geq 0$

$$\delta_2 \geq \lambda_q^2 - q_2 \geq 0$$

$$q_2 + \delta_2 \geq \lambda_q^2 \geq q_2 \geq 0$$

$$\sqrt{(q_2 + \delta_2)} \geq \lambda_q \geq \sqrt{(q_2)} \geq 0$$

$$\sqrt{(q_2 + \delta_2)} - \sqrt{(q_2)} \geq \lambda_q - \sqrt{(q_2)} \geq 0$$

$$\sqrt{(q_2 + \delta_2)} - \sqrt{(q_2)} \geq |\lambda_q - \sqrt{(q_2)}| \geq 0$$

$$\epsilon_3 \equiv |\lambda_q - \sqrt{(q_2)}|$$

$$\sqrt{(q_2 + \delta_2)} - \sqrt{(q_2)} \geq \epsilon_3 \geq 0$$

$$\delta_2 \geq \epsilon_2 \equiv |\lambda_q^2 - q_2| \geq 0$$

If $q_2 \geq \lambda_q^2 \geq 0$

then $|\lambda_q^2 - q_2| = q_2 - \lambda_q^2 \geq 0$

$$\delta_2 \geq q_2 - \lambda_q^2 \geq 0$$

$$\lambda_q^2 \geq q_2 - \delta_2$$

$$q_2 \geq \lambda_q^2 \geq q_2 - \delta_2$$

If $q_2 \geq \delta_2$

then $q_2 - \delta_2 \geq 0$

$$q_2 \geq \lambda_q^2 \geq q_2 - \delta_2 \geq 0$$

$$\sqrt{(q_2)} \geq \lambda_q \geq \sqrt{(q_2 - \delta_2)} \geq 0$$

$$0 \geq \lambda_q - \sqrt{(q_2)} \geq \sqrt{(q_2 - \delta_2)} - \sqrt{(q_2)}$$

$$0 \leq -\lambda_q + \sqrt{(q_2)} \leq -\sqrt{(q_2 - \delta_2)} + \sqrt{(q_2)}$$

$$0 \leq |\lambda_q - \sqrt{(q_2)}| \leq -\sqrt{(q_2 - \delta_2)} + \sqrt{(q_2)}$$

$$\epsilon_3 \equiv |\lambda_q - \sqrt{(q_2)}|$$

$$0 \leq \epsilon_3 \leq -\sqrt{(q_2 - \delta_2)} + \sqrt{(q_2)}$$

$$\delta_2 \geq \epsilon_2 \equiv |\lambda_q^2 - q_2| \geq 0$$

If $q_2 \geq \lambda_q^2 \geq 0$

then $|\lambda_q^2 - q_2| = q_2 - \lambda_q^2 \geq 0$

$$\delta_2 \geq q_2 - \lambda_q^2 \geq 0$$

$$\lambda_q^2 \geq q_2 - \delta_2$$

$$q_2 \geq \lambda_q^2 \geq q_2 - \delta_2$$

If $q_2 \leq \delta_2$

then $q_2 - \delta_2 \leq 0$

$$q_2 \geq \lambda_q^2 \geq 0 \geq q_2 - \delta_2$$

$$q_2 \geq \lambda_q^2 \geq 0$$

$$\sqrt{(q_2)} \geq \lambda_q \geq 0$$

$$0 \geq \lambda_q - \sqrt{(q_2)} \geq -\sqrt{(q_2)}$$

$$0 \leq -\lambda_q + \sqrt{(q_2)} \leq \sqrt{(q_2)}$$

$$0 \leq |\lambda_q - \sqrt{(q_2)}| \leq \sqrt{(q_2)}$$

$$\epsilon_3 \equiv |\lambda_q - \sqrt{(q_2)}|$$

$$0 \leq \epsilon_3 \leq \sqrt{(q_2)}$$

Appendix B

Matric Multiplication of Principal Vectors

Multiplication of a principal vector by an integer power of its matrix. Assume the mth eigenvalue is multiple and the mth eigenvector must be supplemented by principal vectors in order for the vector space to be spanned

$$\cdots, e_m = t_m, t_{m+1}, \cdots, t_{m+s}, e_{m+s+1}, \cdots$$

where

e_i = eigenvector

t_i, $i > m$ = principal vector

s = number of supplemental principal vectors

$e_m = t_m$ = eigenvector from which derived

From the defining relation of a principal vector (5:30)

$$A t_i = t_{i-1} + \lambda_m t_i$$

$$A^2 t_i = A(A t_i)$$

$$= A(t_{i-1} + \lambda_m t_i)$$

$$A t_{i-1} + \lambda_m A t_i$$

$$= (t_{i-2} + \lambda_m t_{i-1}) + \lambda_m (t_{i-1} + \lambda_m t_i)$$

$$= t_{i-2} + 2\lambda_m t_{i-1} + \lambda_m^2 t_i$$

$$A^3 t_i = A(A^2 t_i)$$

$$= A\left(t_{i-2} + 2\lambda_m t_{i-1} + \lambda_m^2 t_i\right)$$

$$= A t_{i-2} + 2\lambda_m A t_{i-1} + \lambda_m^2 A t_i$$

$$= \left(t_{i-3} + \lambda_m t_{i-2}\right) + 2\lambda_m \left(t_{i-2} + \lambda_m t_{i-1}\right)$$

$$+ \lambda_m^2 \left(t_{i-1} + \lambda_m t_i\right)$$

$$= t_{i-3} + 3\lambda_m t_{i-2} + 3\lambda_m^2 t_{i-1} + \lambda_m^3 t_i$$

$$A^4 t_i = A\left(A^3 t_i\right)$$

$$= A\left(t_{i-3} + 3\lambda_m t_{i-2} + 3\lambda_m^2 t_{i-1} + \lambda_m^3 t_i\right)$$

$$= A t_{i-3} + 3\lambda_m A t_{i-2} + 3\lambda_m^2 A t_{i-1} + \lambda_m^3 A t_i$$

$$= \left(t_{i-4} + \lambda_m t_{i-3}\right) + 3\lambda_m \left(t_{i-3} + \lambda_m t_{i-2}\right)$$

$$+ 3\lambda_m^2 \left(t_{i-2} + \lambda_m t_{i-1}\right) + \lambda_m^3 \left(t_{i-1} + \lambda_m t_i\right)$$

$$= t_{i-4} + 4\lambda_m t_{i-3} + 6\lambda_m^2 t_{i-2}$$

$$+ 4\lambda_m^3 t_{i-1} + \lambda_m^4 t_i$$

For the Kth power of matrix A

$$A^K t_i = \sum_j \left[\binom{K}{j} \lambda_m^{K-j} t_{i-j} \right]$$

where

A = matrix

K = the power to which the matrix is raised

t_i = principal vector, $i \neq m$

$j = 0, i-m$ if $\kappa \geq i-m$

$j = 0, \kappa$ if $\kappa \leq i-m$

$t_m = e_m$ = eigenvector from which t_i is derived

$\binom{\kappa}{j} \equiv (\kappa)!/(j)!/(\kappa-j)!$ = binomial coefficient (3:624)

Appendix C

Four by Four Iterative Matric Equations

Implicit Matric Equation, $A x = b$

$$\begin{bmatrix} c & b & a & o \\ b & c & o & a \\ a & o & c & b \\ o & a & b & c \end{bmatrix} \begin{bmatrix} d/(a+b+c) \\ d/(a+b+c) \\ d/(a+b+c) \\ d/(a+b+c) \end{bmatrix} = \begin{bmatrix} d \\ d \\ d \\ d \end{bmatrix}$$

Jacobi Iteration Equation, $x^{(n+1)} = G x^{(n)} + c$

$$x^{(n+1)} = \begin{bmatrix} o & -b/c & -a/c & o \\ -b/c & o & o & -a/c \\ -a/c & o & o & -b/c \\ o & -a/c & -b/c & o \end{bmatrix} x^{(n)} + \begin{bmatrix} 1/c & o & o & o \\ o & 1/c & o & o \\ o & o & 1/c & o \\ o & o & o & 1/c \end{bmatrix} b$$

Gauss-Seidel Iteration Equation, $x^{(n+1)} = G\,x^{(n)} + c$

$$x^{(n+1)} = \begin{bmatrix} 0 & -b/c & -a/c & 0 \\ 0 & b^2/c^2 & ab/c^2 & -a/c \\ 0 & ab/c^2 & a^2/c^2 & -b/c \\ 0 & -2ab^2/c^3 & -2a^2b/c^3 & (a^2+b^2)/c^2 \end{bmatrix} x^{(n)}$$

$$+ \begin{bmatrix} 1/c & 0 & 0 & 0 \\ -b/c^2 & 1/c & 0 & 0 \\ -a/c^2 & 0 & 1/c & 0 \\ 2ab/c^3 & -a/c^2 & -b/c^2 & 1/c \end{bmatrix} b$$

Successive Over-relaxation Iteration Equation, $x^{(n+1)} = G\,x^{(n)} + c$

$$x^{(n+1)} = \begin{bmatrix} (1-w) & w(-b/c) \\ w(1-w)(-b/c) & w^2(b^2/c^2) + (1-w) \\ w^2(1-w)(-a/c) & w^2(ab/c^2) \\ w^2(1-w)(2ab/c^2) & w^2(-2ab^2/c^2) - (1-w)(a/c) \end{bmatrix}$$

$$\begin{matrix} w(-a/c) & 0 \\ w^2(ab/c^2) & w(-a/c) \\ w^2(a^2/c^2) + (1-w) & w(-b/c) \\ w^2(-2a^2b/c^2) - (1-w)(b/c) & w^2(a^2+b^2)/c^2 + (1-w) \end{matrix}\Bigg]\, x^{(n)}$$

$$+ \begin{bmatrix} w/c & 0 & 0 & 0 \\ w^2(-b/c^2) & w/c & 0 & 0 \\ w^2(-a/c^2) & 0 & w/c & 0 \\ w^3(2ab/c^3) & w^2(-a/c^2) & w^2(-b/c^2) & w/c \end{bmatrix} b$$

Jacobi Matrix Eigenvalues (Derived from characteristic equation.)

$$\lambda = \pm \sqrt{(a^2 + b^2 \pm 2ab)} / c$$

Gauss-Seidel Matrix Eigenvalues

$$\lambda = (a^2 + b^2 \pm 2ab) / c^2 \quad , \quad 0 \quad , \quad 0$$

Appendix D

Validation of Analytic Solution

Solve one dimensional heat conduction equation first. (4:32-35).

$$\partial^2 T/\partial x^2 = (1/K)\ \partial T/\partial t$$

$$T = A(x)\,B(t)$$

$$\partial^2 T/\partial x^2 = A''(x)\,B(t)$$

$$\partial T/\partial t = A(x)\,B'(t)$$

$$A''(x)\,B(t) = (1/K)\,A(x)\,B'(t)$$

$$A''(x)/A(x) = (1/K)\,B'(t)/B(t) = -\alpha^2$$

$$A''(x) = -\alpha^2 A(x)$$

$$B'(t) = -\alpha^2 K\,B(t)$$

$$-\alpha^2 K \equiv -\lambda^2$$

$$B'(t) = -\lambda^2 B(t)$$

$$B(t) = e^{-\delta t}$$

$$B'(t) = -\delta e^{-\delta t}$$

$$-\delta e^{-\delta t} = -\lambda^2 e^{-\delta t}$$

$$\delta = \lambda^2$$

$$B(t) = e^{-\lambda^2 t}$$

$$A''(x) = -\alpha^2 A(x)$$

$$A(x) = e^{sx}$$

$$A''(x) = s^2 e^{sx}$$

$$s^2 e^{sx} = -\alpha^2 e^{sx}$$

$$s^2 = -\alpha^2$$

$$A(x) = C_1 e^{i\alpha x} + C_2 e^{-i\alpha x}$$

$$T(x,t) = A(x)\,B(t)$$

$$= \left(C_1 e^{i\alpha x} + C_2 e^{-i\alpha x}\right) e^{-\lambda^2 t}$$

Apply boundary conditions

$$0 = T(0,t)$$

$$= (C_1 + C_2)\, e^{-\lambda^2 t}$$

$$C_1 = -C_2$$

$$T(x,t) = \left(e^{i\alpha x} - e^{-i\alpha x}\right) C_3\, e^{-\lambda^2 t}$$

$$0 = T(\pi,t)$$

$$= \left(e^{i\alpha\pi} - e^{-i\alpha\pi}\right) C_3\, e^{-\lambda^2 t}$$

$$e^{i\alpha\pi} = e^{-i\alpha\pi}$$

$$i\alpha\pi = -i\alpha\pi + in2\pi$$

$$2\alpha = 2n$$

$$\alpha = n$$

$$T_n(x,t) = (e^{inx} - e^{-inx})\, C_3 e^{-\lambda^2 t}$$

$$\sin(nx) = (e^{inx} - e^{-inx})/(2i)$$

$$= C_4 \sin(nx)\, e^{-\lambda^2 t}$$

Apply initial conditions

$$\sin(x) = T_n(x,0)$$

$$= C_4 \sin(nx)$$

$$C_4 = 1$$

$$n = 1$$

$$T(x,t) = \sin(x)\, e^{-\lambda^2 t}$$

$$\alpha^2 K = \lambda^2$$

$$T(x,t) = \sin(x)\, e^{-\alpha^2 K t}$$

$$\alpha = n$$

$$T(x,t) = \sin(x)\, e^{-n^2 K t}$$

$$n = 1$$

$$T(x,t) = \sin(x)\, e^{-Kt}$$

Solve two dimensional problem using one dimensional solutions (2:33-34).

$$T(x,y,t) = T(x,t)\,T(y,t)$$

$$= \sin(x)\sin(y)\,e^{-2\kappa t}$$

Boundary conditions are

$$T(0,y,t) = T(\pi,y,t)$$

$$= T(x,0,t) = T(x,\pi,t) = 0$$

Initial conditions are

$$T(x,y,0) = \sin(x)\sin(y)$$

Domain

$$0 \le x \le \pi$$

$$0 \le y \le \pi$$

$$0 \le t$$

Validation of solution

$$T(x,y,t) = \sin(x)\sin(y)\,e^{-2\kappa t}$$

$$\partial T/\partial x = \cos(x)\sin(y)\,e^{-2\kappa t}$$

$$\partial^2 T/\partial x^2 = -\sin(x)\sin(y)\,e^{-2\kappa t}$$

$$\partial^2 T/\partial y^2 = -\sin(x)\sin(y)\,e^{-2\kappa t}$$

$$\partial T/\partial t = -2\kappa\sin(x)\sin(y)\,e^{-2\kappa t}$$

$$\partial^2 T/\partial x^2 + \partial^2 T/\partial y^2 = -2\sin(x)\sin(y)\,e^{-2\kappa t}$$

$$-2 \sin(x) \sin(y) e^{-2\kappa t} = (1/\kappa)\, \partial T/\partial t$$

$$T(x, y, 0) = \sin(x) \sin(y)$$

$$T(0, y, t) = \sin(0) \sin(y) e^{-2\kappa t} = 0$$

$$T(x, 0, t) = \sin(x) \sin(0) e^{-2\kappa t} = 0$$

$$T(\pi, y, t) = \sin(\pi) \sin(y) e^{-2\kappa t} = 0$$

$$T(x, \pi, t) = \sin(x) \sin(\pi) e^{-2\kappa t} = 0$$

Appendix E

Computer Code- Spectral Radius, Single Iteration

```
1=      PROGRAM SPCRAD
2=C     SPECTRAL RADIUS AS A FUNCTION OF NODE SPACING
3=      IMPLICIT DOUBLEPRECISION(A-Z)
4=      INTEGER N,NN,NNN,LIMIT,ITLR
5=      PARAMETER(dT=1,DIFF=1.0,LIMIT=30,INTRVL=1,EL=1.E-15,NNN=9,NN=9
6=     . ,PI=3.14159265358979323846264338327950288419716939)
7=      DIMENSION VX((NNN-1)*(NNN-1)),VY((NNN-1)*(NNN-1))
8=     . ,SOR(3,NN),JACOBI(3,NN),3,NN)
9=      CHARACTER F1*22,F2*51
10=     F1='(/3(T2,6A14/),T58,A14)'
11=     F2='(/T2,2114,3E14.5,I14,2E14.5/T2,114,E28.5/T2,E42.5/)'
12=     WRITE(*,'(''1'')')
13=     WRITE(*,*)'dT=',dT
14=     WRITE(*,*)'DIFF=',DIFF
15=     WRITE(*,*)'LIMIT=',LIMIT
16=     WRITE(*,*)'INTRVL=',INTRVL
17=     WRITE(*,*)'EL=',EL
18=     WRITE(*,*)'NN=',NN
19=     WRITE(*,*)'NN=',NN
20=     WRITE(*,F1)'NODAL','INTERIOR','MATRIX','JACOBI','JACOBI'
21=    . ,'NUMBER','SOR','SOR','DENSITY','MODE','COEFF','SPECTRAL'
22=    . ,'SPECTRAL','MATRIX','SPECTRAL','ACCEL','dX','dY','NUMBER'
23=    . ,'A','B','C','RADIUS','ITLR','RADIUS','CONST','ERROR'
24=C    N=2 YIELDS A SINGLE ELEMENT, ZERO JACOBI ITERATION MATRIX,
25=C    ITS ONLY EIGENVALUE EQUALS ZERO,
26=C    THUS ITS SPECTRAL RADIUS EQUALS ZERO.
27=     DO 1 N=3,NN
28=        DO 2 I=3,NN
29=C          NUMBER OF INTERIOR NODES
30=           NN=(N-1)*(N-1)
31=C          SET MESH SPACING
32=           dX=PI/N
33=           dY=PI/N
34=C          MATRIX COEFFICIENTS
35=           A=-DIFF*dT/dX**2
```

```
36=               B=-DIFF*RT/RY**2
37=               C=1+2*DIFF*RT*(RX**(-2)+RY**(-2))
38=C              SPECTRAL RADIUS OF JACOBI ITERATION MATRIX
39=               CALL SPCTRL(A,B,C,RK,N,VX,VY,LIMIT,INTRVL
40=              .,EL,JACOBI(M,N),E,ITER)
41=C              DETERMINE SOR ACCELERATION CONSTANT
42=               ALPHA=2/(1+SQRT(1-(JACOBI(M,N))**2))
43=C              SPECTRAL RADIUS OF SOR ITERATION MATRIX
44=               SOR(M,N)=ALPHA-1
45=               WRITE(*,F2)M,RN,A,JACOBI(M,N),E,ITER,SOR(M,N),ALPHA,B,C
46=2           CONTINUE
47=1        CONTINUE
48=         OPEN(1,FILE='DATA')
49=         REWIND 1
50=         WRITE(1,*)JACOBI
51=         WRITE(1,*)SOR
52=         END
```

```
53=       SUBROUTINE SPCTRL(A,B,C,MM,M,X,Y,LIMIT,INTRVL,EL,L,E,K)
54=C      SPECTRAL RADIUS OF JACOBI ITERATION MATRIX
55=       IMPLICIT DOUBLEPRECISION(A-Z)
56=       INTEGER MM,M,I,J,K,LIMIT,MODI,IM,INTRVL
57=       DIMENSION X(*),Y(*)
58=C      INITIAL TRIAL VECTOR
59=       DO 1 I=1,MM
60=          X(1)=1
61=1      CONTINUE
62=C      LIMIT NUMBER OF ITERATIONS
63=       DO 2 K=1,LIMIT
64=          YM=0
65=C         INCREMENT MATRIX ROWS AND NEW ITERATED VECTOR ELEMENTS
66=          DO 3 I=1,MM
67=             Y(1)=0
68=             MODI=MOD(I,M-1)
69=C            INCREMENT MATRIX COLUMNS AND OLD ITERATED VECTOR ELEMENTS
70=C            SKIP MATRIX ELEMENTS THAT EQUAL ZERO
71=             DO 4 J=1,MM
72=C               STRICTLY LOWER TRIANGULAR
73=                IF(J.EQ.I-1.AND.MODI.NE.1)THEN
74=                   Y(I)=Y(I)-B/C*X(J)
75=                ELSEIF(J.EQ.I-M+1)THEN
76=                   Y(I)=Y(I)-A/C*X(J)
77=C               STRICTLY UPPER TRIANGULAR
78=                ELSEIF(J.EQ.I+1.AND.MODI.NE.0)THEN
79=                   Y(I)=Y(I)-B/C*X(J)
80=                ELSEIF(J.EQ.I+M-1)THEN
81=                   Y(I)=Y(I)-A/C*X(J)
82=                ENDIF
83=4            CONTINUE
84=C            MAXIMUM MODULUS VECTOR ELEMENT
85=             IF(ABS(Y(I)).GT.YM)THEN
86=                IM=I
87=                YM=ABS(Y(I))
```

```
 88=            ENDIF
 89=3         CONTINUE
 90=C         EIGENVALUE AND EIGENVALUE ERROR
 91=          IF(MOD(K,INTRVL).EQ.0)THEN
 92=            A0=0
 93=            A1=0
 94=            A2=0
 95=            DO 5 I=1,NN
 96=              A0=A0+X(I)**2
 97=              A1=A1+X(I)*Y(I)
 98=              A2=A2+Y(I)**2
 99=5           CONTINUE
100=C           RAYLEIGH EIGENVALUE APPROXIMATION
101=            L=A2/A1
102=C           ABS USED TO AVOID NEGATIVE ARGUEMENTS
103=C           CAUSED BY TRUNCATION ERROR
104=            E=SQRT(ABS(L**2-A2/A0))
105=            IF(E.LT.EL)GOTO 6
106=          ENDIF
107=C         LIMIT TRIAL VECTOR ELEMENT MAGNETUDE
108=          DO 7 I=1,NN
109=            X(I)=Y(I)/Y(IM)
110=7         CONTINUE
111=2       CONTINUE
112=        K=LIMIT
113=6       END
```

```
      PROGRAM SPCRAD
C     SPECTRAL RADIUS AS A FUNCTION OF NODE SPACING
      IMPLICIT DOUBLEPRECISION(A-Z)
      INTEGER I,II,J,JJ,NN,MN,LIMIT,INTRVL,ITER
      PARAMETER(MT=1,DIFF=1.0,LIMIT=30,INTRVL=1,EL=1E-15,MM=9,NN=9
     .,PI=3.14159265358979323846264338327950288419720D0)
      DIMENSION VX((MM-1)*(NN-1)),VY((MM-1)*(NN-1))
     .,VZ((MM-1)*(NN-1))
     .,SOR(3:MM,3:NN),JACOBI(3:MM,3:NN)
      CHARACTER F1*22,F2*51
      F1='(/3(T2,8A14/),T56,A14)'
      F2='(/T2,2I14,3E14.5,I14,2E14.5/T2,I14,E28.5/T2,E42.5/)'
      WRITE(*,'(''1'')')
      WRITE(*,*)'MT=',MT
      WRITE(*,*)'DIFF=',DIFF
      WRITE(*,*)'LIMIT=',LIMIT
      WRITE(*,*)'INTRVL=',INTRVL
      WRITE(*,*)'EL=',EL
      WRITE(*,*)'MM=',MM
      WRITE(*,*)'NN=',NN
      WRITE(*,F1)'NODAL','INTERIOR','MATRIX','JACOBI','JACOBI'
     .,'NUMBER','SOR','SOR','DENSITY','NODE','COEFF','SPECTRAL'
     .,'SPECTRAL','MATRIX','SPECTRAL','ACCEL','I BY N','NUMBER'
     .,'A,B,C','RADIUS','RADIUS','ITER','RADIUS','CONST','ERROR'
C     M=N=2 YIELDS A SINGLE ELEMENT, ZERO JACOBI ITERATION MATRIX,
C     ITS ONLY EIGENVALUE EQUALS ZERO,
C     THUS ITS SPECTRAL RADIUS EQUALS ZERO
      DO 1 M=3,MM
         DO 2 N=3,NN
C           NUMBER OF INTERIOR NODES
            MN=(M-1)*(N-1)
C           SET MESH SPACING
            HX=PI/M
            HY=PI/N
C           MATRIX COEFFICIENTS
```

```
36=                A=-DIFF*HT/HX**2
37=                B=-DIFF*HT/HY**2
38=                C=1+2*DIFF*HT*(HX**(-2)+HY**(-2))
39=C               SPECTRAL RADIUS OF JACOBI ITERATION MATRIX
40=                CALL SPCTRL(A,B,C,AN,E,VX,VY,VZ,LIMIT,INTRVL
41=               .,EL,JACOBI(M,N),E,ITER)
42=C               DETERMINE SOR ACCELERATION CONSTANT
43=                ALPHA=2/(1+SQRT(1-(JACOBI(M,N))**2))
44=C               SPECTRAL RADIUS OF SOR ITERATION MATRIX
45=                SOR(M,N)=ALPHA-1
46=                WRITE(*,F2)M,AN,A,JACOBI(M,N),E,ITER,SOR(M,N),ALPHA,N,B,C
47=2            CONTINUE
48=1         CONTINUE
49=          OPEN(1,FILE='DATA')
50=          REWIND 1
51=          WRITE(1,*)JACOBI
52=          WRITE(1,*)SOR
53=          END
```

```
54=      SUBROUTINE SPCTRL(A,B,C,MN,N,X,Y,Z,LIMIT,INTRVL,EL,L,E,K)
55=C     SPECTRAL RADIUS OF JACOBI ITERATION MATRIX
56=      IMPLICIT DOUBLEPRECISION(A-Z)
57=      INTEGER MN,N,I,K,LIMIT,IM,INTRVL
58=      DIMENSION X(*),Y(*),Z(*)
59=C     INITIAL TRIAL VECTOR
60=      DO 1 I=1,MN
61=         X(I)=1
62=1     CONTINUE
63=C     LIMIT NUMBER OF ITERATIONS
64=      DO 2 K=1,LIMIT/2
65=         CALL ITER(X,Y,A,B,C,MN,N,IM)
66=         CALL ITER(Y,Z,A,B,C,MN,N,IM)
67=C        EIGENVALUE AND EIGENVALUE ERROR
68=         IF(MOD(K,INTRVL).EQ.0)THEN
69=            M0=0
70=            M1=0
71=            M2=0
72=            DO 5 I=1,MN
73=               M0=M0+X(I)**2
74=               M1=M1+X(I)*Z(I)
75=               M2=M2+Z(I)**2
76=5           CONTINUE
77=C           RAYLEIGH EIGENVALUE APPROXIMATION
78=            LL=M2/M1
79=            L=SQRT(LL)
80=C           ABS USED TO AVOID NEGATIVE ARGUEMENTS
81=C           CAUSED BY TRUNCATION ERROR
82=            EE=SQRT(ABS(LL**2-M2/M0))
83=            E1=SQRT(LL+EE)-SQRT(LL)
84=            IF(LL.LE.EE)THEN
85=               E2=SQRT(LL)
86=            ELSEIF(LL.GE.EE)THEN
87=               E2=-SQRT(LL-EE)+SQRT(LL)
88=            ENDIF
```

```
89=               E=MAX(E1,E2)
90=               IF(E.LT.EL)GOTO 6
91=            ENDIF
92=C           LIMIT TRIAL VECTOR ELEMENT MAGNETUDE
93=            DO 7 I=1,MN
94=               X(I)=Z(I)/Z(IM)
95=7           CONTINUE
96=2        CONTINUE
97=         K=LIMIT
98=6        END
```

```
 99=      SUBROUTINE ITER(X,Y,A,B,C,NN,N,IM)
100=C     ONE ITERATION OF THE POWER METHOD, MATRIC MULTIPLICATION
101=      IMPLICIT DOUBLEPRECISION(A-Z)
102=      INTEGER NN,N,IM,I,J,MODI
103=      DIMENSION X(*),Y(*)
104=      YM=0
105=C     INCREMENT MATRIX ROWS AND NEW ITERATED VECTOR ELEMENTS
106=      DO 3 I=1,NN
107=         Y(I)=0
108=         MODI=MOD(I,N-1)
109=C        INCREMENT MATRIX COLUMNS AND OLD ITERATED VECTOR ELEMENTS
110=C        SKIP MATRIX ELEMENTS THAT EQUAL ZERO
111=         DO 4 J=1,NN
112=C           STRICTLY LOWER TRIANGULAR
113=            IF(J.EQ.I-1.AND.MODI.NE.1)THEN
114=               Y(I)=Y(I)-B/C*X(J)
115=            ELSEIF(J.EQ.I-N+1)THEN
116=               Y(I)=Y(I)-A/C*X(J)
117=C           STRICTLY UPPER TRIANGULAR
118=            ELSEIF(J.EQ.I+1.AND.MODI.NE.0)THEN
119=               Y(I)=Y(I)-B/C*X(J)
120=            ELSEIF(J.EQ.I+N-1)THEN
121=               Y(I)=Y(I)-A/C*X(J)
122=            ENDIF
123=4        CONTINUE
124=C        MAXIMUM MODULUS VECTOR ELEMENT
125=         IF(ABS(Y(I)).GT.YM)THEN
126=            IM=I
127=            YM=ABS(Y(I))
128=         ENDIF
129=3     CONTINUE
130=      END
```

Appendix G

Computer Code- Iterative Error

```
      PROGRAM ERROR
      MATRIX  SOLUTION VERSES ITERATED SOLUTION
      IMPLICIT DOUBLEPRECISION(A-Z)
      INTEGER NODE,A,B,I,J,TIME,NTIME,NEW,OLD,ITER,NITER,ANODE
     . ,K,LIMIT,INTRVL,NE,JCBITR,METHOD,NOW,LAST
      LOGICAL L2,L3,L4,L5,L6,L66,L8
      PARAMETER(DT=1,DIFF=1.0,NTIME=1,A=3,K=3,ANELEM=25
     . ,NITER=30,LIMIT=30,INTRVL=1,EL=1E-15,METHOD=1,ZERO=0
     . ,PI=3.14159265358979323346264338327950288419720D0)
      DIMENSION Y((A-1)*(B-1)),ITEMP(2,(A-1)*(B-1)),P((A-1)*(B-1))
     . ,ERROR(0:NITER),TEMP((B-1)*(B-1)),L((B-1)*(B-1))
     . ,VX((B-1)*(B-1)),VY((B-1)*(B-1)),ITHORA(0:NITER)
     . ,ERORA(0:NITER),AT(2,(B-1)*(B-1)),ATX((B-1)*(A-1))
     . ,L1((B-1)*(B-1)),L2((B-1)*(B-1)),L2A(0:NITER)
      DI(1:NITER),EE(1:NITER)
      CHARACTER F1*10,F2*15,F3*31,F4*12 F5*10,F6*15,F7*14
     . ,F8*7,F9*15,F10*3,F11*14,F12*14,F13*12,F14*12
     . ,F15*12,F16*12
     . F8='(''1''')'
      F10='(/)'
      WRITE(*,F8)
      NUMBER OF INTERIOR NODES
      NE=(B-1)*(A-1)
      SET MESH SPACING
      AX=PI/A
      AY=PI/B
      MATRIX COEFFICIENTS
      A=-DIFF*DT/AX**2
      B=-DIFF*DT/AY**2
      C=1+2*DIFF*DT*(AX**(-2)+AY**(-2))
      SPECTRAL RADIUS OF JACOBI ITERATION MATRIX
      CALL SPCTRL((A),(B),(C),(NE),E,VX,VY,LIMIT,INTRVL
     . ,LL,JACOBI,JCBERR,JCBITR)
      SPECTRAL RADIUS OF GAUSS-SEIDEL ITERATION MATRIX
      GAUSS=JACOBI**2
```

```
C       OPTIMIZE SOR ACCELERATION CONSTANT
C       TO MINIMIZE SPECTRAL RADIUS
        AOPT=2/(1+SQRT(1-JACOBI**2))
C       SET ACCELERATION CONSTANT
        ALPHA=AOPT
C       SPECTRAL RADIUS OF SOR ITERATION MATRIX
        IF(ALPHA.GE.AOPT.AND.ALPHA.LT.2)THEN
           SOR=ALPHA-1
        ELSEIF(ALPHA.LE.AOPT.AND.ALPHA.GT.0)THEN
           SOR=(ALPHA*JACOBI/2+SQRT((ALPHA*JACOBI/2)**2-ALPHA+1))**2
        ENDIF
        LHH=N.EQ.4.AND.M.EQ.3
C       INFINITE NORM OF ITERATION MATRIX
        IF(LHH)THEN
           INFNOR=ALPHA-1+2*(-ALPHA*B/C)
        ENDIF
        WRITE(* ,10)
        WRITE(*,*)'NODAL SPACING = ',A,' BY ',B
        WRITE(*,*)'NUMBER OF INTERIOR NODES=',MN
        WRITE(*,*)'MT=',MT
        WRITE(*,*)'MA=',MA
        WRITE(*,*)'NY=',NY
        WRITE(*,*)'DIFF=',DIFF
        WRITE(*,*)'A=',A
        WRITE(*,*)'B=',B
        WRITE(*,*)'C=',C
        WRITE(*,*)'JACOBI SPECTRAL RADIUS = ',JACOBI
        WRITE(*,*)'JACOBI SPECTRAL RADIUS ERROR LIMIT=',JCBERR
        WRITE(*,*)'NUMBER OF JACOBI SPECTRAL RADIUS ITERATIONS=',JCBITR
        WRITE(*,*)'GAUSS-SEIDEL SPECTRAL RADIUS=',GAUSS
        WRITE(*,*)'OPTIMUM SOR ACCELERATION CONSTANT=',AOPT
        WRITE(*,*)'SOR ACCELERATION CONSTANT=',ALPHA
        WRITE(* *)'SOR SPECTRAL RADIUS = ',SOR
        IF(LHH)THEN
           WRITE(*,*)'SOR INFINITE NORM=',INFNOR
```

```
 71=        ENDIF
 72=        WRITE(*,*)'METHOD=',METHOD
 73=C       INCREMENT TIME
 74=        NEW=1
 75=        NOW=1
 76=        LH=.....(.LT.3.OR...GT.5)
 77=        DO 6 TIME=0,ATIME
 78=C          INITIALIZE ROWS
 79=           A(.)=
 80=           B(.)=0
 81=           TEMP(.)=0
 82=           DO 7 NODE=1,...
 83=              IF(LH)THEN
 84=C                PHYSICAL SOLUTION UNKNOWN
 85=C                PT=0 B(.)=-.1 E=-11
 86=                 TEMP(NODE)=0
 87=              ELSE
 88=C                SUBSCRIPT INTERIOR NODES. NODE=J+(I-1)*(I-1)
 89=                 J=MOD(NODE-1,I-1)+1
 90=                 I=(NODE-J)/(I-1)+1
 91=C                PHYSICAL SOLUTION
 92=                 TEMP(NODE)=SIN(PI*I)*SIN(PI*J)*EXP(-2*DIFF*PI*PI*TIME)
 93=              ENDIF
 94=              TNORM=MAX(TNORM,ABS(TEMP(NODE)))
 95=7          CONTINUE
 96=C          MATRIX SOLUTION
 97=C          SAVE IT FOR TWO TIME STEPS
 98=           IF(NOW.EQ.1)THEN
 99=              NOW=2
100=              LAST=1
101=           ELSE
102=              NOW=1
103=              LAST=2
104=           ENDIF
105=           IF(LH)THEN
```

```
106=            CALL MTRXI(MT,A,B,C,NN,I,TIME,MTIME,NOW,LAST)
107=          ELSEIF(TIME.EQ.0)THEN
108=            DO 10 NODE=1,NN
109=              MT(NOW,NODE)=TEMP(NODE)
110=10          CONTINUE
111=          ELSEIF(N.EQ.N)THEN
112=            IF(N.EQ.3)THEN
113=              DO 12 NODE=1,NN
114=                MT(NOW,NODE)=MT(LAST,NODE)/(A+B+C)
115=12            CONTINUE
116=            ELSEIF(N.EQ.4)THEN
117=              CALL MTRX4(MTX,MT,A,B,C,LAST)
118=              DO 13 NODE=1,NN
119=                MT(NOW,NODE)=MTX(NODE)
120=13            CONTINUE
121=            ELSEIF(N.EQ.5)THEN
122=               CALL MTRX5(MTX,MT,A,B,C,LAST)
123=               DO 14 NODE=1,NN
124=                 MT(NOW,NODE)=MTX(NODE)
125=14            CONTINUE
126=            ENDIF
127=          ENDIF
128=          IF(TIME.NE.0)THEN
129=            WRITE(*,'(/T2,A,16)')'TIME STEP=',TIME
130=            DO 16 NODE=1,NN
131=              MTN=MAX(MTN,ABS(MT(NOW,NODE)))
132=C             PHYSICAL SOLUTION MINUS MATRIX SOLUTION
133=              E1(NODE)=TEMP(NODE)-MT(NOW,NODE)
134=              E1N=MAX(E1N,ABS(E1(NODE)))
135=C             ITERATED SOLUTION
136=              IF(METHOD.EQ.1)THEN
137=                ITEMP(NEW,NODE)=MT(LAST,NODE)
138=              ELSEIF(METHOD.EQ.2)THEN
139=                ITEMP(NEW,NODE)=1
140=               ENDIF
```

```
141=16          CONTINUE
142=            DO 5 ITER=0,NITER
143=C               INITIALIZE NORMS
144=                ITNORM(ITER)=0
145=                E2N(ITER)=0
146=                ENORM(ITER)=0
147=                DO 9 NODE=1,NN
148=C                   ITERATED SOLUTION NORM
149=                    ITNORM(ITER)=MAX(ITNORM(ITER),ABS(ITEMP(NEW,NODE)))
150=C                   MATRIX SOLUTION MINUS ITERATED SOLUTION
151=                    E2(NODE)=RT(NOW,NODE)-ITEMP(NEW,NODE)
152=                    E2N(ITER)=MAX(E2N(ITER),ABS(E2(NODE)))
153=C                   PHYSICAL SOLUTION MINUS ITERATED SOLUTION
154=                    E(NODE)=TEMP(NODE)-ITEMP(NEW,NODE)
155=                    ENORM(ITER)=MAX(ENORM(ITER),ABS(E(NODE)))
156=9               CONTINUE
157=C               SAVE ITEMP FOR ONLY TWO ITERATIONS
158=                IF(NEW.EQ.1)THEN
159=                    NEW=2
160=                    OLD=1
161=                ELSE
162=                    NEW=1
163=                    OLD=2
164=                ENDIF
165=C               INITIALIZE NORMS
166=                DNORM(ITER)=0
167=C               INCREMENT MATRIX ROWS
168=C               ARE ELEMENTS OF NEW ITERATED VECTOR
169=                DO 4 NODE=1,NN
170=C                   SUMMATIONS IN SOR ALGORITHM
171=C                   SKIP MATRIX ELEMENTS THAT EQUAL ZERO
172=                    MNODE=MOD(NODE,N-1)
173=                    SUM1=0
174=                    SUM2=0
175=C                   INCREMENT MATRIX COLUMNS
```

```
176=                  DO 3 K=1 ,NN
177=C                    STRICTLY LOWER TRIANGULAR ELEMENTS
178=                     IF(K.EQ.NODE-1.AND.NNODE.NE.1)THEN
179=                        SUM1=SUM1+B*ITEMP(NEW,K)
180=                     ELSEIF(K.EQ.NODE-NN+1)THEN
181=                        SUM1=SUM1+A*ITEMP(NEW,K)
182=C                    DIAGONAL ELEMENTS
183=                      ELSEIF(K.EQ.NODE)THEN
184=                        SUM2=SUM2+C*ITEMP(OLD,K)
185=C                    STRICTLY UPPER TRIANGULAR ELEMENTS
186=                      ELSEIF(K.EQ.NODE+1.AND.NNODE.NE.0)THEN
187=                        SUM2=SUM2+B*ITEMP(OLD,K)
188=                      ELSEIF(K.EQ.NODE+NN-1)THEN
189=                        SUM2=SUM2+A*ITEMP(OLD,K)
190=                     ENDIF
191=3                 CONTINUE
192=C                 DISPLACEMENT
193=                   D(NODE)=-ALPHA/C*(-RT(LAST,NODE)+SUM1+SUM2)
194=C                 D(ITER)=X(ITER+1)-X(ITER)
195=                  DNORM(ITER)=MAX(DNORM(ITER),ABS(D(NODE)))
196=C                 ITERATED SOLUTION
197=                  ITEMP(NEW,NODE)=ITEMP(OLD,NODE)+D(NODE)
198=4              CONTINUE
199=C              AVOID WRITING TOO MANY NODES
200=              IF(NN.LE.MAXLIN)THEN
201=                 F5='(/I2,A,I4)'
202=                  F14='(/T20,4A13/)'
203=                  F13='(T20,4E13.5)'
204=                  WRITE(*,F5)'ITERATION=',ITER
205=                 WRITE(*,F14)'ITNORM','E2R','RTR','DNORM'
206=                 WRITE(*,F13)ITNORM(ITER),E2R(ITER),RTR,DNORM(ITER)
207=                 WRITE(*,F14)'ITEMP','E2','RT','D'
208=                 WRITE(*,F13)(ITEMP(OLD,NODE),E2(NODE)
209=                 ,RT(NOW,NODE),D(NODE),NODE=1,NN)
210=              ENDIF
```

```
211=5           CONTINUE
212=             F6='(//T2,A4,7A13/)'
213=             F7='(T2,I4,7E13.5)'
214=             WRITE(*,F6)'ITER','ITNORM','E2N','ATN'
215=            .,'E1N','TNORM','ENORM','DNORM'
216=            DO 1 ITER=0,MITER
217=               WRITE(*,F7)ITER,ITNORM(ITER),E2N(ITER),ATN
218=            .     ,E1N,TNORM,ENORM(ITER),DNORM(ITER)
219=1           CONTINUE
220=C           QUANTITIES DERIVED FROM IT,L2,AT,E1,T,L,D
221=            F9='(//T2,A4,6A17/)'
222=            WRITE(*,F9)'ITER','1/(1-K0(I))','K0(I)'
223=            .,'K1(I)/(1-K1(I))','K1(I)'
224=            .,'K(L(I)/L(I-1))','K(D(I)/D(I-1))'
225=            DO 6 ITER=0,MITER
226=               EDO=E2N(ITER)/ENORM(ITER)
227=               KO=1-1/EDO
228=               IF(ITER.GE.1)THEN
229=                  EO1=E2N(ITER)/ENORM(ITER-1)
230=                  K1=EO1/(1+EO1)
231=                   DD(ITER)=DNORM(ITER)/DNORM(ITER-1)
232=                   EE(ITER)=E2N(ITER)/E2N(ITER-1)
233=                  F11='(T2,I4,6E17.5)'
234=                  WRITE(*,F11)ITER,EDO,KO
235=            .           ,EO1,K1,EE(ITER),DD(ITER)
236=               ELSE
237=                  F12='(T2,I4,2E17.5)'
238=                  WRITE(*,F12)ITER,EDO,KO
239=               ENDIF
240=6           CONTINUE
241=            DO 2 ITER=0,MITER
242=               RENORM=ENORM(ITER)/ITNORM(ITER)
243=               RDNORM=DNORM(ITER)/ITNORM(ITER)
244=                RE1N=E1N/ITNORM(ITER)
245=               RE2N=E2N(ITER)/ITNORM(ITER)
```

```
246=                F1='(/T2,A,13)'
247=                F2='(T2,A,T9,E11.5)'
248=                F3='(T2,A,I9,E11.5,T26,A,T33,E11.5)'
249=                WRITE(*,F1)'ITERATION=',ITER
250=                WRITE(*,F2)'ITNORM=',ITNORM(ITER)
251=                WRITE(*,F3)'E2N=',E2N(ITER),'RE2N=',RE2N
252=                WRITE(*,F2)'ATK=',ATK
253=                WRITE(*,F3)'E1N=',E1N,'RE1N=',RE1N
254=                WRITE(*,F2)'TNORM=',TNORM
255=                WRITE(*,F3)'ENORM=',ENORM(ITER),'RENORM=',RENORM
256=                WRITE(*,F3)'DNORM=',DNORM(ITER),'RDNORM=',RDNORM
257=C           VARIATIONS IN CONTRACTIVE CONSTANT
258=C           CONTRACTIVE-CONSTANT EXISTANCE CONDITIONS
259=C           LE(I+1) (E(I),E(I+1))
260=            L2=ITER.LE.NITER-1
261=C           LE(I) (E(I-1),E(I))
262=             L3=ITER.GE.1
263=C           DD(I+1) (D(I),D(I+1))
264=             L4=ITER.LE.NITER-1
265=C           DD(I) (D(I-1),D(I))
266=            L5=ITER.GE.1
267=C           ERROR-NORM-LIMIT EXISTANCE CONDITIONS
268=C           ELI (D(I-1))
269=            L6=ITER.NE.0
270=            F4='(/T17,9A13/)'
271=            WRITE(*,F4)'K','1/(1-K)','K/(1-K)','ELO','ELI','ELI'
272=           ,'RELO','RELI','RELI'
273=            IF(L6)THEN
274=               IF(L2)THEN
275=                   CALL LIMIT1('K(E(I+1)/E(I))',LE(ITER+1)
276=                  ,(DNORM(ITER)),(DNORM(ITER-1)),(DNORM(0))
277=                  ,(ITER),(ITNORM(ITER)))
278=                ENDIF
279=               IF(L3)THEN
280=                  CALL LIMIT1('K(E(I)/E(I-1))',LE(ITER)
```

```
281=      .                ,(DNORM(ITER)),(DNORM(ITER-1)),(DNORM(0))
282=      .                ,(ITER),(ITNORM(ITER)))
283=                   ENDIF
284=                   IF(L4)THEN
285=                      CALL LIMIT1('R(D(I+1)/D(I))',DD(ITER+1)
286=      .                ,(DNORM(ITER)),(DNORM(ITER-1)),DNORM(0)
287=      .                 (ITER),(ITNORM(ITER)))
288=      -            ENDIF
289=                   IF(L5)THEN
290=                      CALL LIMIT1('R(D(1)/D(I-1))',(DD(ITER))
291=      .                ,(DNORM(ITER)),(DNORM(ITER-1)),(DNORM(0))
292=      .                ,(ITER),(ITNORM(ITER)))
293=                   ENDIF
294=                   CALL LIMIT1('JACOBI',(JACOBI)
295=      .            ,(DNORM(ITER)),(DNORM(ITER-1)),(DNORM(0))
296=      .            ,(ITER),(ITNORM(ITER)))
297=                   CALL LIMIT1('GAUSS',(GAUSS)
298=      .            ,(DNORM(ITER)),(DNORM(ITER-1)),(DNORM(0))
299=      .            ,(ITER),(ITNORM(ITER)))
300=                    CALL LIMIT1('SOR',(SOR)
301=      .            ,(DNORM(ITER)),(DNORM(ITER-1)),(DNORM(0))
302=      .            ,(ITER),(ITNORM(ITER)))
303=                    CALL LIMIT1('ZERO',(ZERO)
304=      .            ,(DNORM(ITER)),(DNORM(ITER-1)),(DNORM(0))
305=      .            ,(ITER),(ITNORM(ITER)))
306=                ELSE
307=                   IF(L2)THEN
308=                      CALL LIMIT2('R(E(I+1)/E(I))',EE(1)
309=      .                ,DNORM(0),ITNORM(0))
310=                   ENDIF
311=                   IF(L4)THEN
312=                      CALL LIMIT2('R(D(I+1)/D(I))',DD(1)
313=      .                ,DNORM(0),ITNORM(0))
314=                   ENDIF
315=                   CALL LIMIT2('JACOBI',(JACOBI),(DNORM(0))
```

```
316=     .                ,(ITNORM(0)))
317=                      CALL LIMIT2('GAUSS',(GAUSS),(DNORM(0))
318=     .                ,(ITNORM(0)))
319=                      CALL LIMIT2('SOR',SOR,DNORM(0)
320=     .                ,(ITNORM(0)))
321=                      CALL LIMIT2('ZERO',(ZERO),(DNORM(0))
322=     .                ,(ITNORM(0)))
323=                   ENDIF
324=2               CONTINUE
325=               F15='(//T2,3A30/)'
326=               F16='(I2,3E30.27)'
327=               WRITE(*,F15)'ITEMP','HT','E2'
328=               WRITE(*,F16)(ITEMP(OLD,NODE),HT(NOW,NODE)
329=     .         ,E2(NODE),NODE=1,NN)
330=            ENDIF
331=6        CONTINUE
332=C        TRANSFER DATA FOR PLOTTING
333=        OPEN(1,FILE='DATA')
334=        REWIND 1
335=        WRITE(1,*)SOR
336=        WRITE(1,*)E2N
337=        WRITE(1,*)DNORM
338=        END
```

```
      SUBROUTINE SPCTRL(A,B,C,MN,N,X,Y,LIMIT,INTRVL,EL,L,E,K)
C     SPECTRAL RADIUS OF JACOBI ITERATION MATRIX
      IMPLICIT DOUBLEPRECISION(A-Z)
      INTEGER MN,N,I,J,K,LIMIT,MODI,IM,INTRVL
      CHARACTER F2*10,F3*16
      DIMENSION X(*),Y(*)
      F2='(T2,3A36/)'
      F3='(T2,I36,2E36.27)'
      WRITE(*,F2)'ITERATION','JACOBI SPECTRAL RADIUS'
     .,'ERROR LIMIT'
C     INITIAL TRIAL VECTOR
      DO 1 I=1,MN
         X(I)=1
1     CONTINUE
C     LIMIT NUMBER OF ITERATIONS
      DO 2 K=1,LIMIT
         YM=0
C        INCREMENT MATRIX ROWS AND NEW ITERATED VECTOR ELEMENTS
         DO 3 I=1,MN
            Y(I)=0
            MODI=MOD(I,N-1)
C           INCREMENT MATRIX COLUMNS AND OLD ITERATED VECTOR ELEMENTS
C           SKIP MATRIX ELEMENTS THAT EQUAL ZERO
            DO 4 J=1,MN
C              STRICTLY LOWER TRIANGULAR
               IF(J.EQ.I-1.AND.MODI.NE.1)THEN
                  Y(I)=Y(I)-B/C*X(J)
               ELSEIF(J.EQ.I-N+1)THEN
                  Y(I)=Y(I)-A/C*X(J)
C              STRICTLY UPPER TRIANGULAR
               ELSEIF(J.EQ.I+1.AND.MODI.NE.0)THEN
                  Y(I)=Y(I)-B/C*X(J)
               ELSEIF(J.EQ.I+N-1)THEN
                  Y(I)=Y(I)-A/C*X(J)
               ENDIF
```

```
374=4            CONTINUE
375=C            MAXIMUM MODULUS VECTOR ELEMENT
376=             IF(ABS(Y(I)).GT.YM)THEN
377=                IM=I
378=                YM=ABS(Y(I))
379=             ENDIF
380=3         CONTINUE
381=C         EIGENVALUE AND EIGENVALUE ERROR
382=          IF(MOD(K,INTRVL).EQ.0)THEN
383=             M0=0
384=             M1=0
385=             M2=0
386=             DO 5 I=1,MN
387=                M0=M0+X(I)**2
388=                M1=M1+X(I)*Y(I)
389=                M2=M2+Y(I)**2
390=5            CONTINUE
391=C            RAYLEIGH EIGENVALUE APPROXIMATION
392=             E=M2/M1
393=C            ABS USED TO AVOID NEGATIVE ARGUMENTS
394=C            CAUSED BY TRUNCATION ERROR
395=             E=SQRT(ABS(E**2-M2/M0))
396=             WRITE(*,F3)K,E
397=             IF(E.LT.EL)GOTO 6
398=          ENDIF
399=C         LIMIT TRIAL VECTOR ELEMENT MAGNITUDE
400=          DO 7 I=1,MN
401=             X(I)=Y(I)/Y(IM)
402=7         CONTINUE
403=2      CONTINUE
404=       K=LIMIT
405=6      END
```

```
      SUBROUTINE LIMIT1(KTITLE,K,DNI,DNH,DNO,ITER,ITN)
C     ERROR NORM LIMITS
      IMPLICIT DOUBLEPRECISION(A-Z)
      INTEGER ITER
      CHARACTER FORM*17,KTITLE*(*)
      C0=1/(1-K)
      C1=K/(1-K)
      CI=K**ITER/(1-K)
      ELO=C0*DNI
      EL1=C1*DNH
      ELI=CI*DNO
      RELO=ELO/ITN
      REL1=EL1/ITN
      RELI=ELI/ITN
      FORM='(T2,A,T17,9E13.5)'
      WRITE(*,FORM)KTITLE,K,C0,C1,ELO,EL1,ELI,RELO,REL1,RELI
      END
```

```
      SUBROUTINE LIMIT2(KTITLE,K,DX,ITN)
C     ERROR NORM LIMITS
      IMPLICIT DOUBLEPRECISION(A-Z)
      CHARACTER FORM*37,KTITLE*(*)
      C=1/(1-K)
      EL=C*DX
      REL=EL/ITN
      FORM='(T2,A,T17,2E13.5,TR13,2(L13.5,L26.5))'
      WRITE(*,FORM)KTITLE,K,C,EL,EL,REL,REL
      END
```

```
433=      SUBROUTINE MTRX4(MT,Y,A,B,C,LAST)
434=C     MATRIX SOLUTION FOR 4 BY 4 NODE SPACING
435=      IMPLICIT DOUBLEPRECISION(A-Z)
436=      INTEGER LAST
437=      DIMENSION MT(*),Y(2,*)
438=      D=Y(LAST,1)
439=      E=Y(LAST,2)
440=      F=Y(LAST,5)
441=      DD=C**3-8*A**2*C
442=      DG=D*(C**2-4*A**2)-2*A*(E*C-F*A)
443=      DH=C*(E*C-F*A)-2*A*C*D
444=      DI=-4*A*(C*E-2*A*D)+F*(C**2-4*A**2)
445=      G=DG/DD
446=      H=DH/DD
447=      I=DI/DD
448=      MT(1)=G
449=      MT(2)=H
450=      MT(3)=G
451=      MT(4)=H
452=      MT(5)=I
453=      MT(6)=H
454=      MT(7)=G
455=      MT(8)=H
456=      MT(9)=G
457=      END
```

```
      SUBROUTINE MTRX5(MT,Y,A,B,C,LAST)
C     MATRIX SOLUTION FOR 5 BY 5 NODE SPACING
      IMPLICIT DOUBLEPRECISION(A-Z)
      INTEGER LAST
      DIMENSION MT(*),Y(2,*)
      D=Y(LAST,1)
      E=Y(LAST,2)
      F=Y(LAST,6)
      DD=C**2*(3*A+C)-2*A**2*(2*A+C)
      DC=C*D*(3*A+C)-2*A*(E*(2*A+C)-A*F)
      DA=C*(E*(2*A+C)-F*A)-A*D*(2*A+C)
      D1=C*(F*(A+C)-2*A*E)-2*A**2*(F-D)
      C=DC/DD
      A=DA/DD
      B=D1/DD
      MT(1)=C
      MT(2)=A
      MT(3)=B
      MT(4)=C
      MT(5)=A
      MT(6)=1
      MT(7)=1
      MT(8)=A
      MT(9)=A
      MT(10)=1
      MT(11)=1
      MT(12)=A
      MT(13)=C
      MT(14)=A
      MT(15)=A
      MT(16)=C
      END
```

```
490=      SUBROUTINE MTRX1(X,A,B,C,MM,N,TIME,NTIME,NOW,LAST)
491=C     GENERATE MATRIX SOLUTIONS
492=      IMPLICIT DOUBLEPRECISION(A-Z)
493=      INTEGER I,J,NTIME,TIME,I,OLD,NEW,LAST,NOW,MM,N
494=      DIMENSION X(2,*)
495=      OLD=1
496=C     FINAL TIME, MATRIX SOLUTION
497=      DO 3 I=1,MM
498=         X(OLD,I)=1
499=3     CONTINUE
500=      DO 4 T=NTIME,TIME,-1
501=         IF(OLD.EQ.1)THEN
502=            OLD=2
503=            NEW=1
504=         ELSE
505=            OLD=1
506=            NEW=2
507=         ENDIF
508=C        INCREMENT MATRIX ROWS
509=C        AND UNKNOWN VECTOR ELEMENTS
510=         DO 1 I=1,MM
511=            X(OLD,I)=0
512=            MODI=MOD(I,N-1)
513=C           INCREMENT MATRIX COLUMNS
514=C           AND KNOWN VECTOR ELEMENTS
515=C           SKIP MATRIX ELEMENTS THAT EQUAL ZERO
516=            DO 2 J=1,MM
517=C              STRICTLY LOWER TRIANGULAR ELEMENTS
518=               IF(J.EQ.I-1.AND.MODI.NE.1)THEN
519=                  X(OLD,I)=X(OLD,I)+B*X(NEW,J)
520=               ELSEIF(J.EQ.I-N+1)THEN
521=                  X(OLD,I)=X(OLD,I)+A*X(NEW,J)
522=C              DIAGONAL ELEMENTS
523=               ELSEIF(J.EQ.I)THEN
524=                  X(OLD,I)=X(OLD,I)+C*X(NEW,J)
```

```
525=C                 STRICTLY UPPER TRIANGULAR ELEMENTS
526=                  ELSEIF(J.EQ.I+1.AND.MOD1.NE.0)THEN
527=                     X(OLD,I)=X(OLD,I)+B*X(NEW,J)
528=                  ELSEIF(J.EQ.I+N-1)THEN
529=                     X(OLD,I)=X(OLD,I)+A*X(NEW,J)
530=                  ENDIF
531=2              CONTINUE
532=1           CONTINUE
533=4        CONTINUE
534=         NOW=NEW
535=         LAST=OLD
536=         END
```

Appendix II

Iterative Error Approximations

3 by 3 Nodal Density

Number of interior nodes = 4
Time interval between matric solutions = 1
Diffusion coefficient = 1
Computer precision = 27 decimal digits
Number of spectral radius iterations = 30
Jacobi spectral radius = .39242
Gauss-Seidel spectral radius = .15399
SOR optimum spectral radius = .041782
Number of iterations = 30

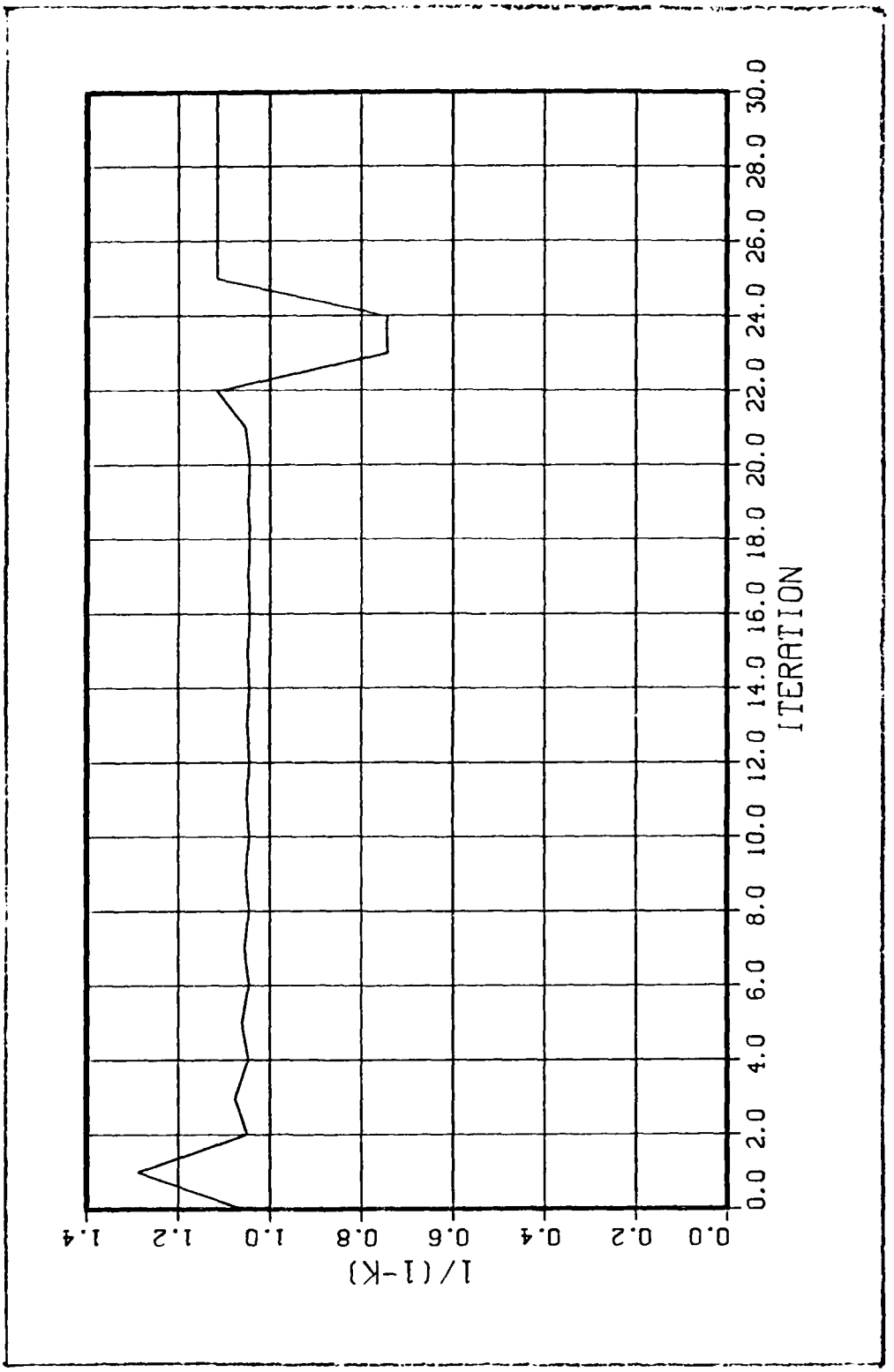


Figure 22. Error Norm to Displacement Norm Ratio, Error Limit Method 1, 3 x 3 Nodal Density

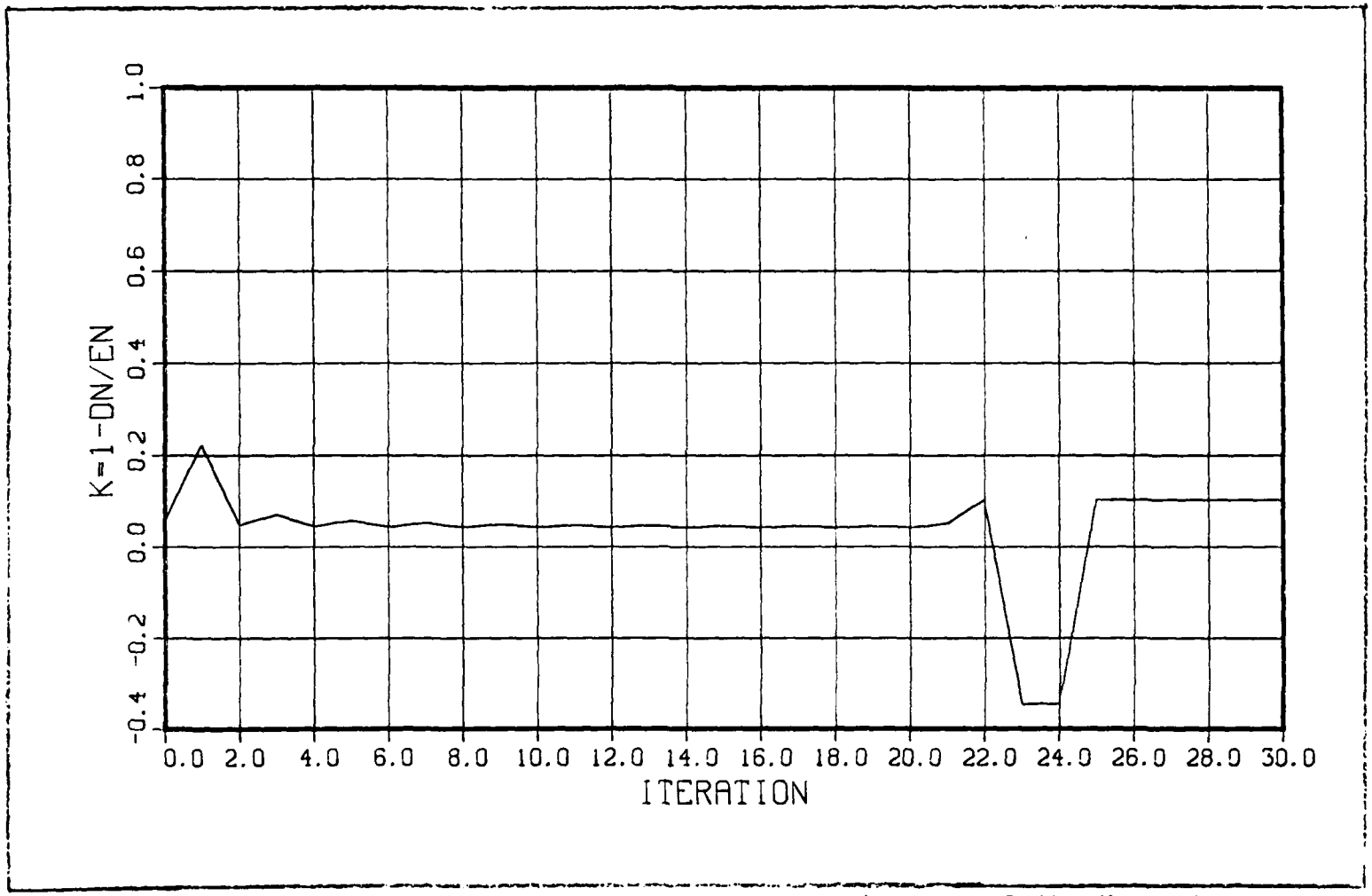


Figure 23. Parameter Required for Error Limit Method 1 to Equal the Error Norm, 3 x 3 Nodal Density

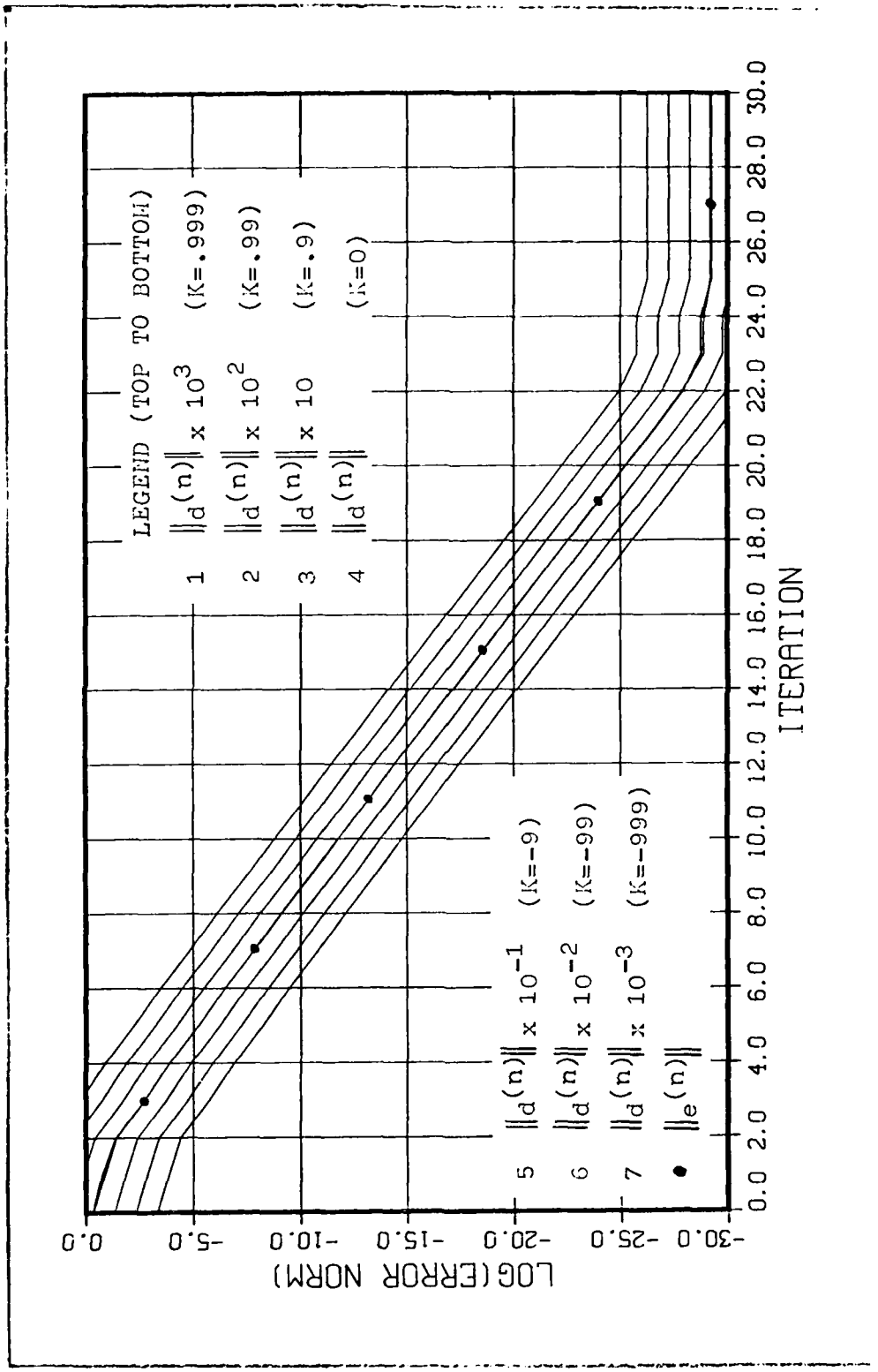


Figure 24. Order of Magnitude Comparison between Error Norm and Displacement Norm, Error Limit Method 1, 3 x 3 Nodal Density

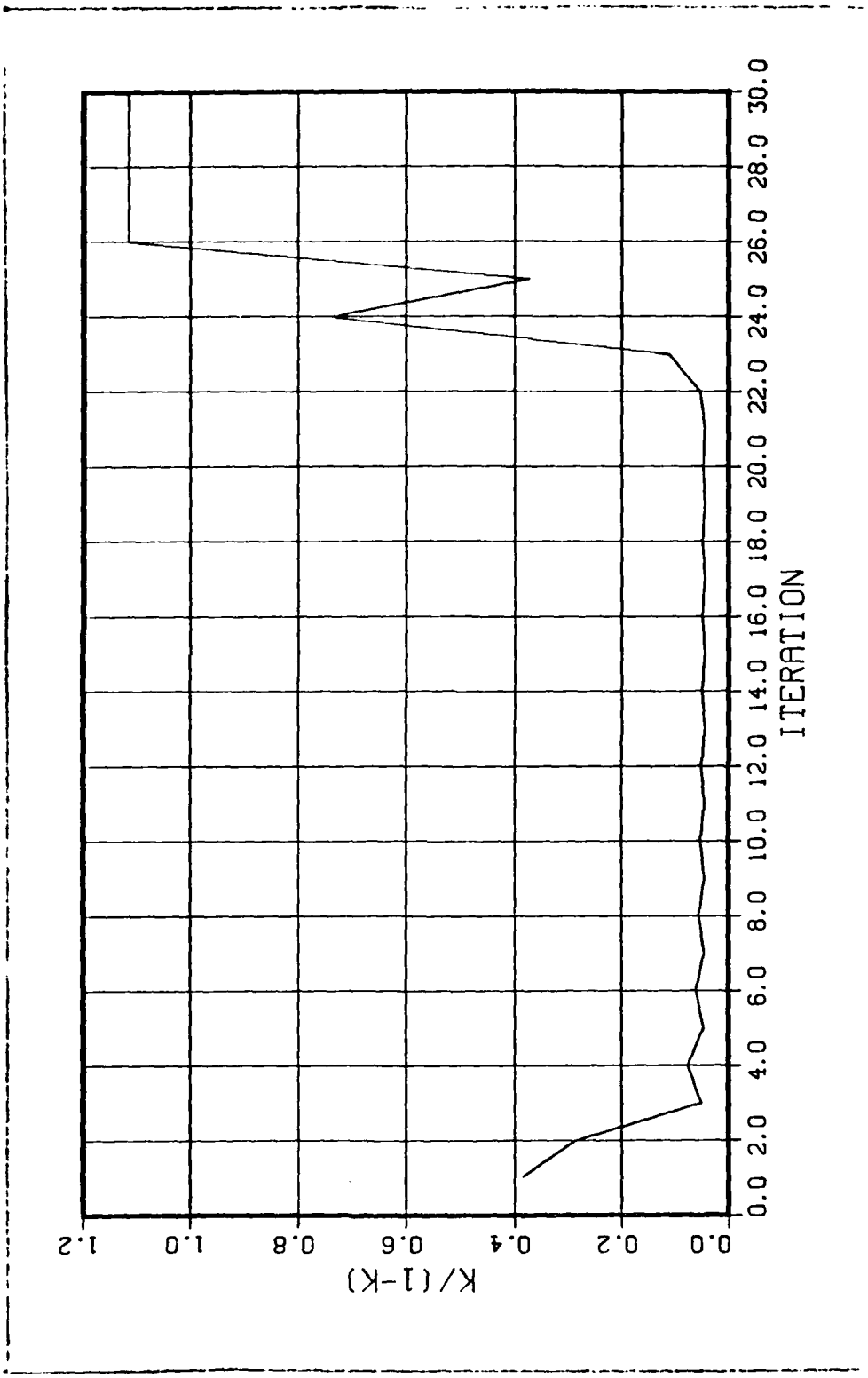


Figure 25. Error Norm to Displacement Norm Ratio, Error Limit Method 2, 3 x 3 Nodal Density

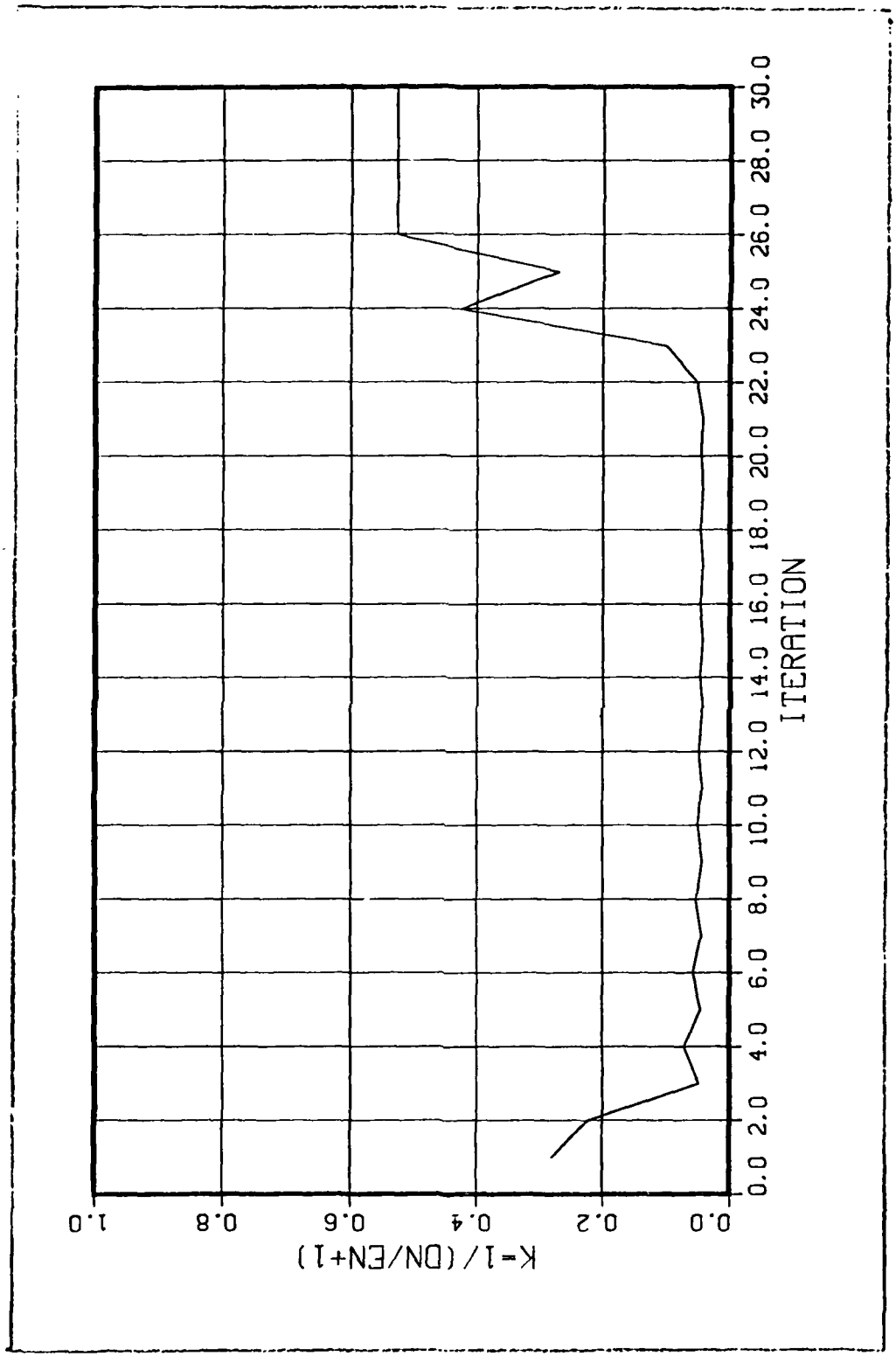


Figure 26. Parameter Required for Error Limit Method 2 to Equal the Error Norm, 3 x 3 Nodal Density

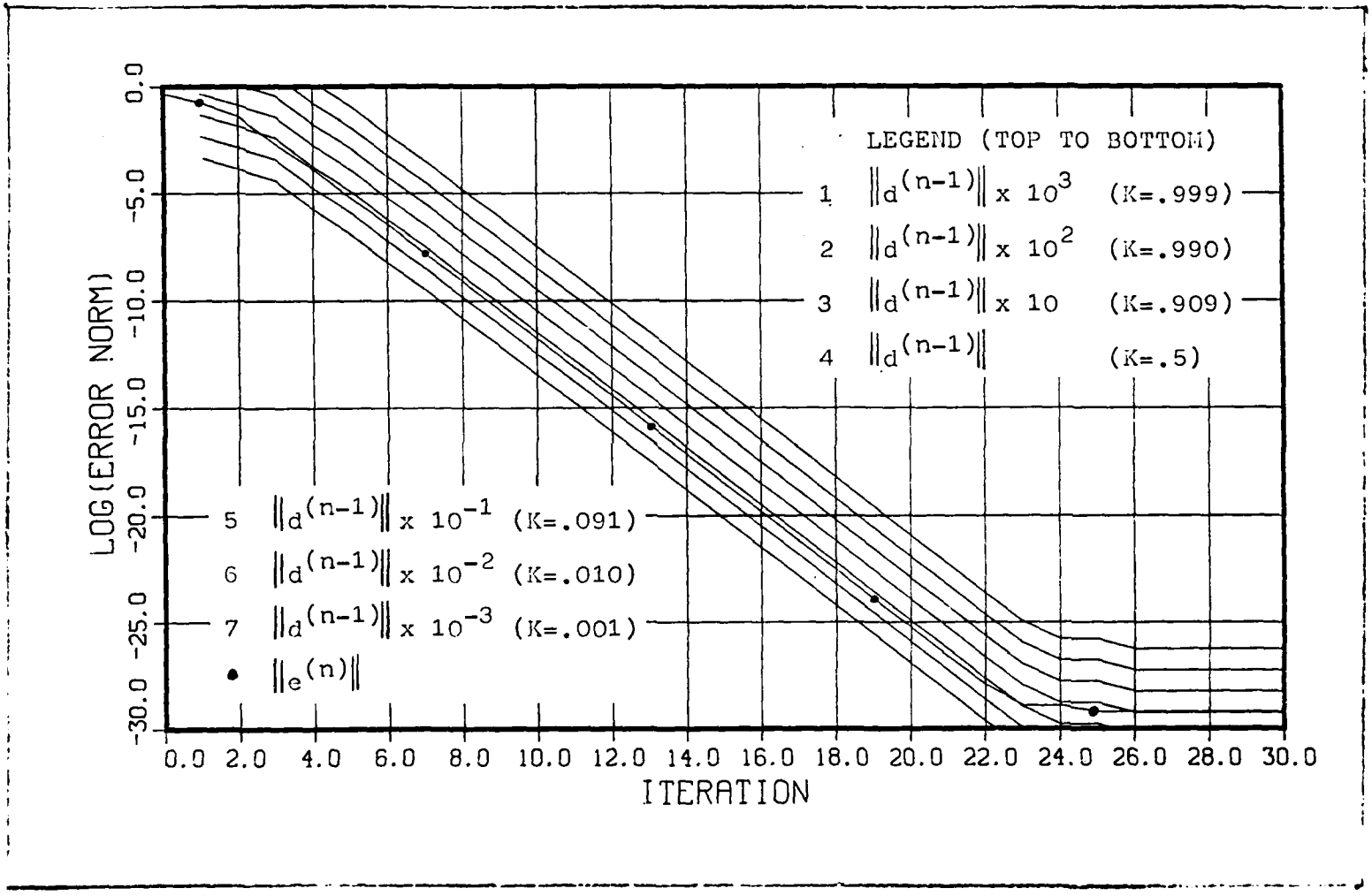


Figure 27. Order of Magnitude Comparison between Error Norm and Displacement Norm, Error Limit Method 2, 3 x 3 Nodal Density

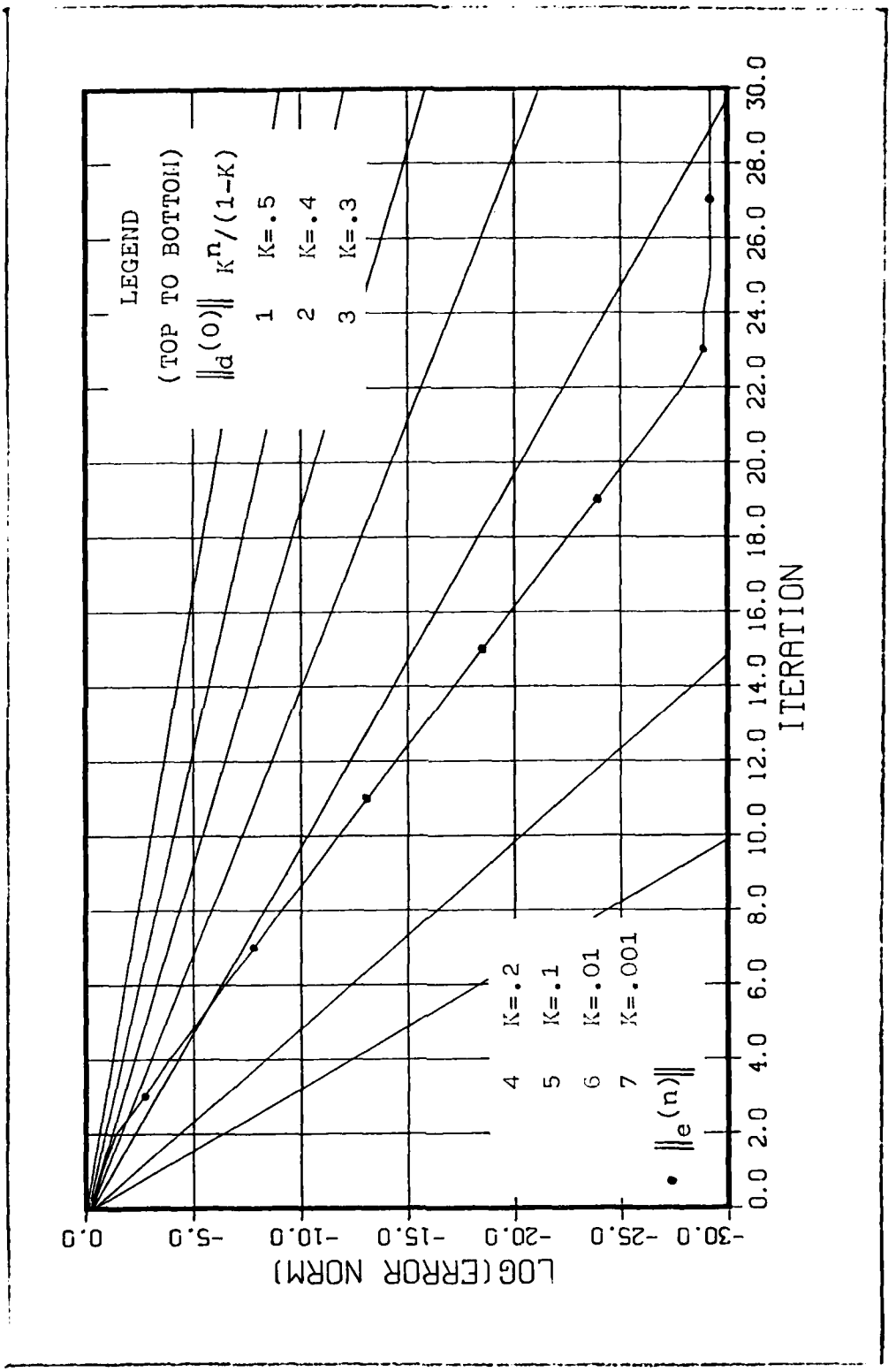


Figure 28. Comparison of Error Norm to Error Limit Method 3, 3 x 3 Nodal Density

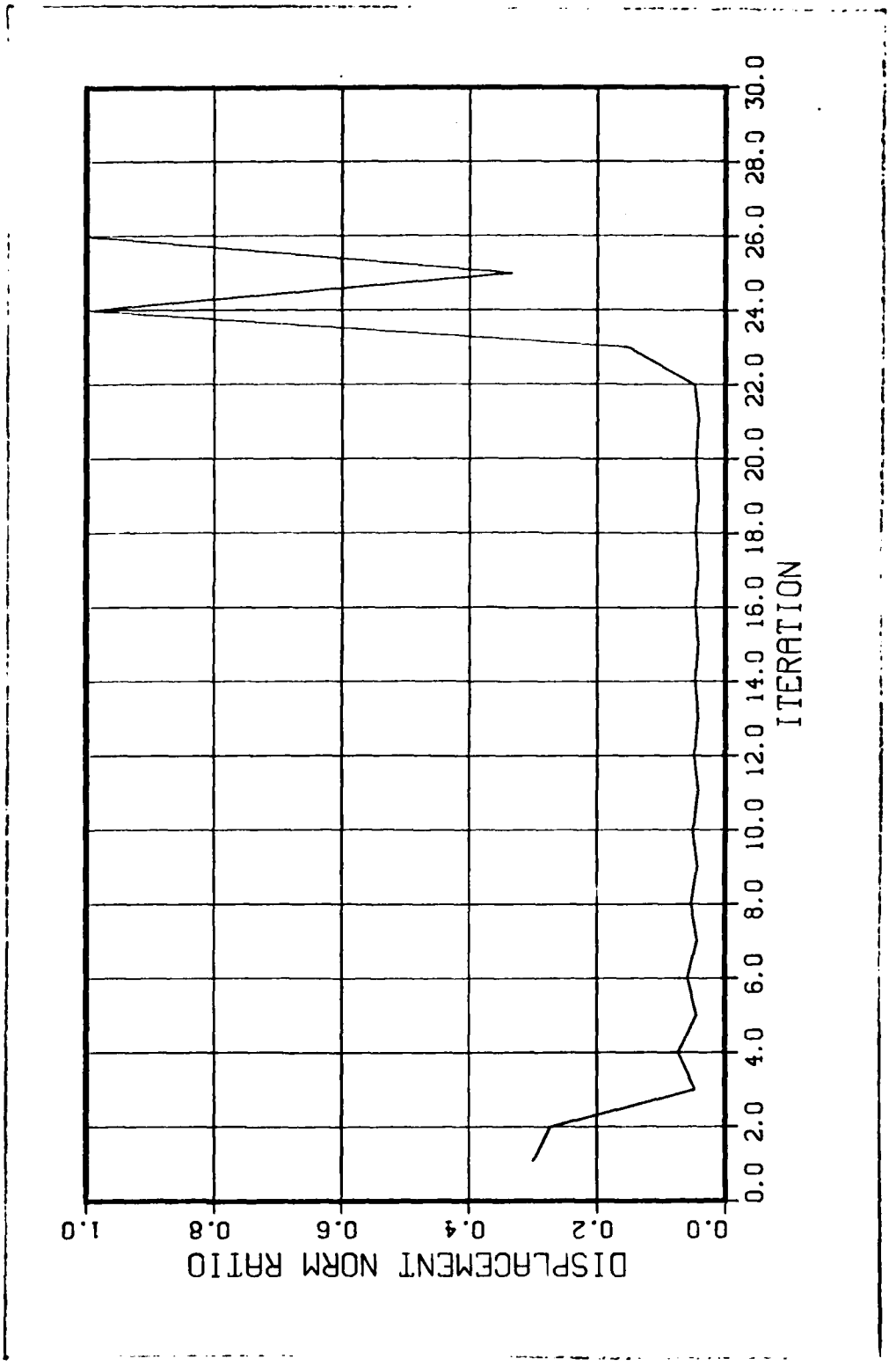


Figure 29. Displacement Norm Ratio, 3 x 3 Modal Density

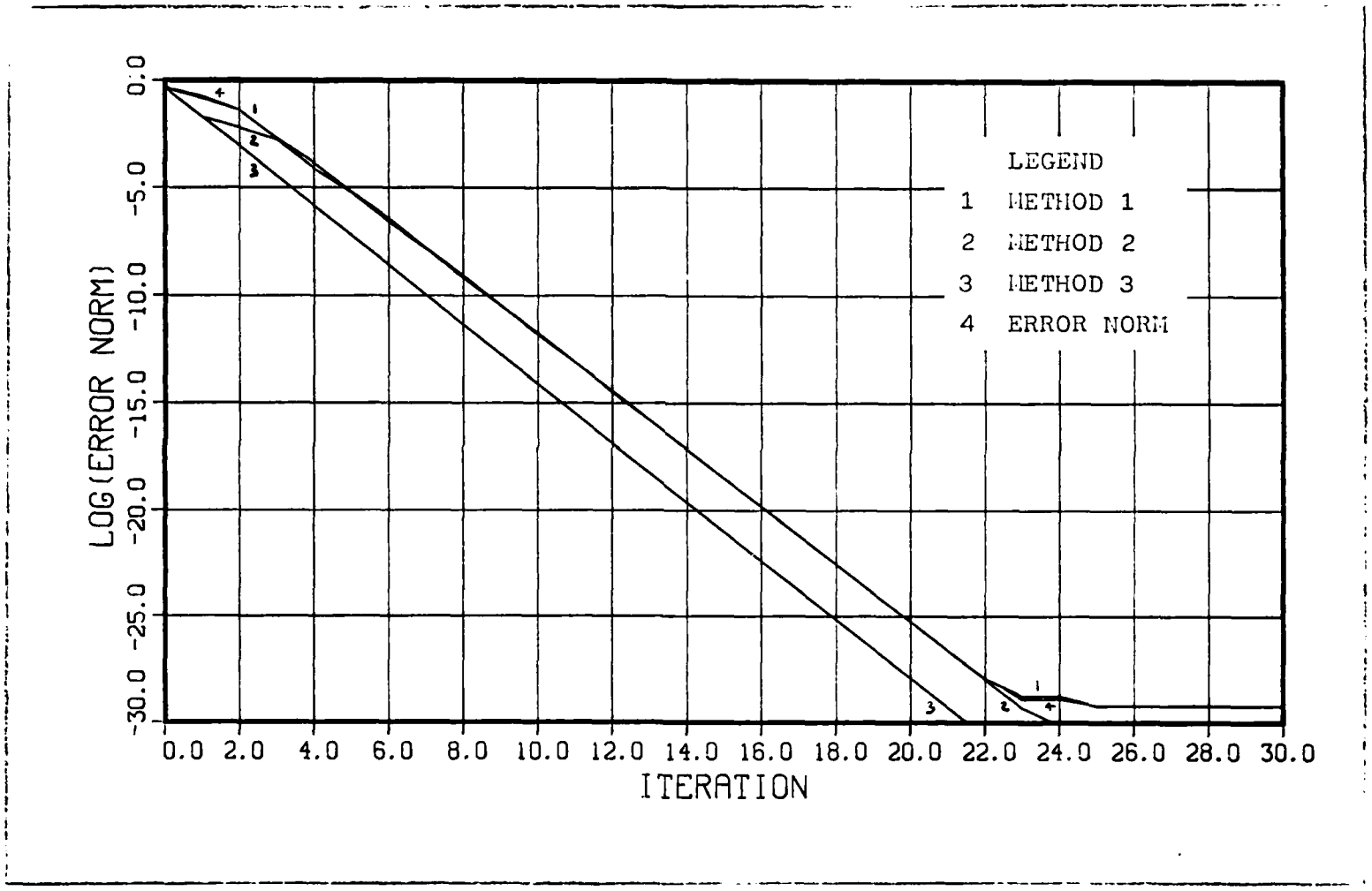


Figure 30. Three Error Limit Methods Compared to Error Norm, K = SOR Optimum Spectral Radius, 3 x 3 Nodal Density

Appendix I

Iterative Error Approximations
20 by 20 Modal Density

Number of interior nodes = 361
Time interval between matric solutions = 10,000,000
Diffusion coefficient = 1
Computer precision = 13 decimal digits
Number of spectral radius iterations = 30
Jacobi spectral radius = .98715
Gauss-Seidel spectral radius =.97447
SOR optimum spectral radius = .72446
Number of iterations = 200

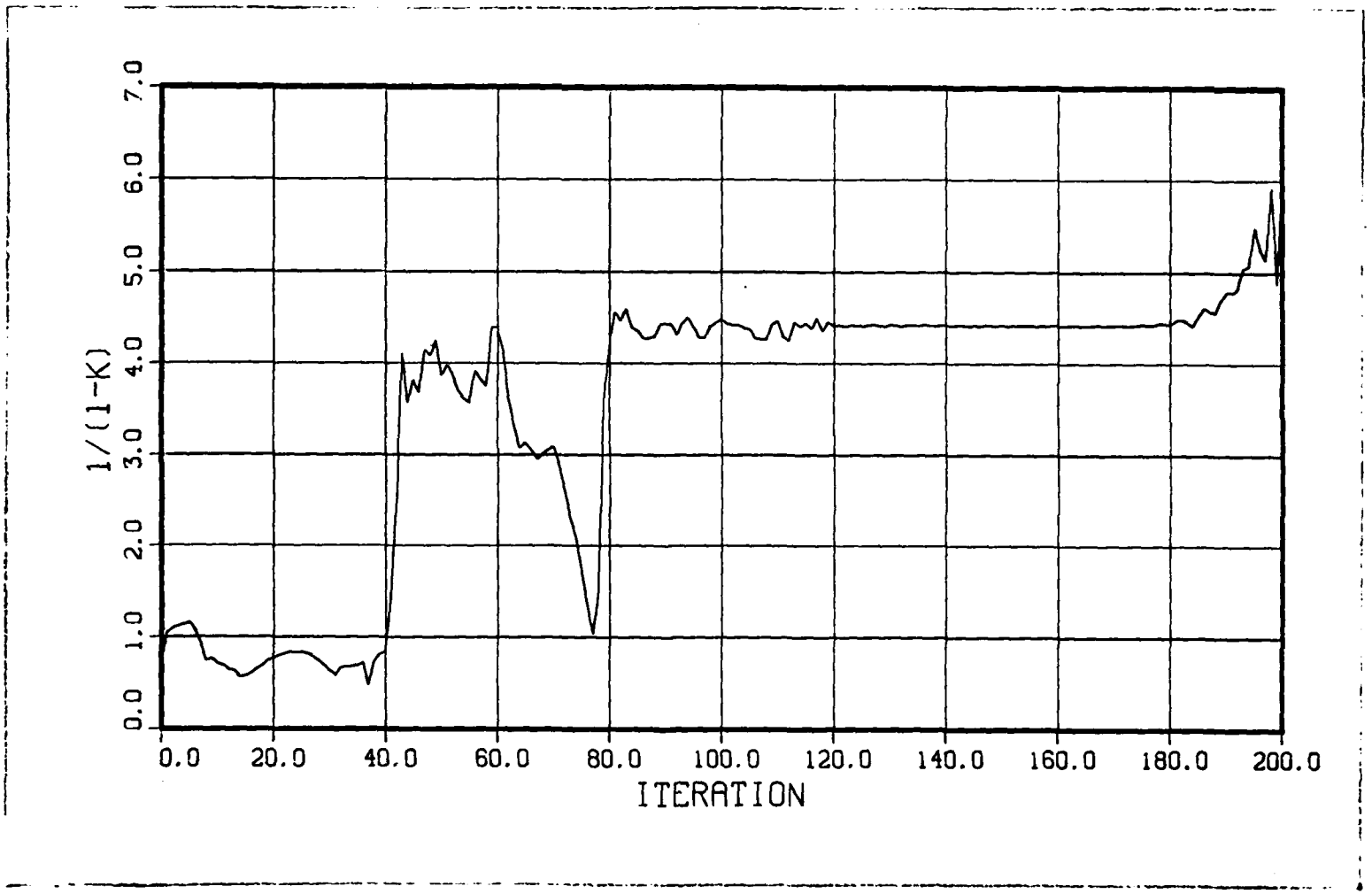


Figure 31. Error Norm to Displacement Norm Ratio, Error Limit Method 1, 20 x 20 Nodal Density

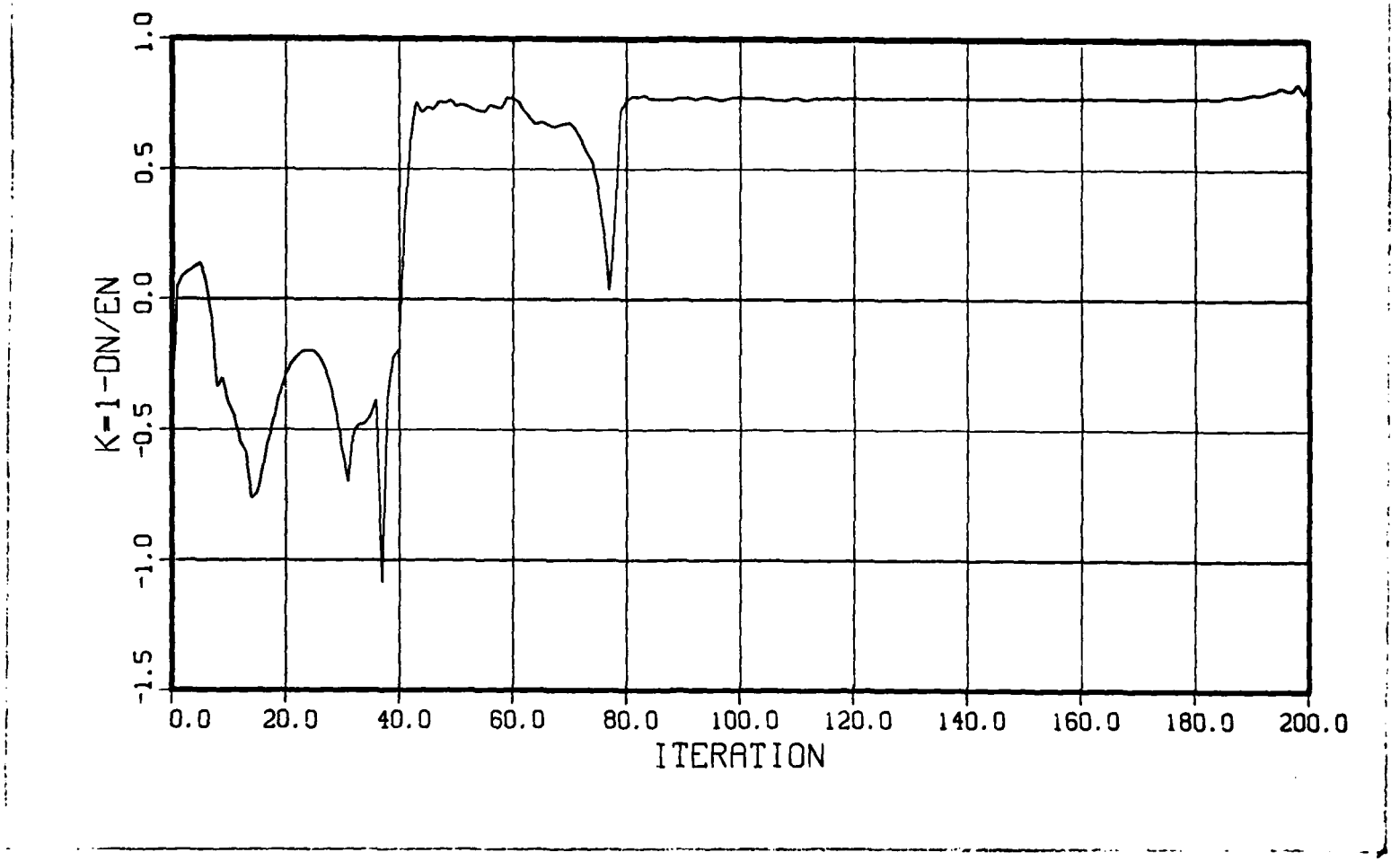


Figure 32. Parameter Required for Error Limit Method 1 to Equal the Error Norm, 20 x 20 Nodal Density

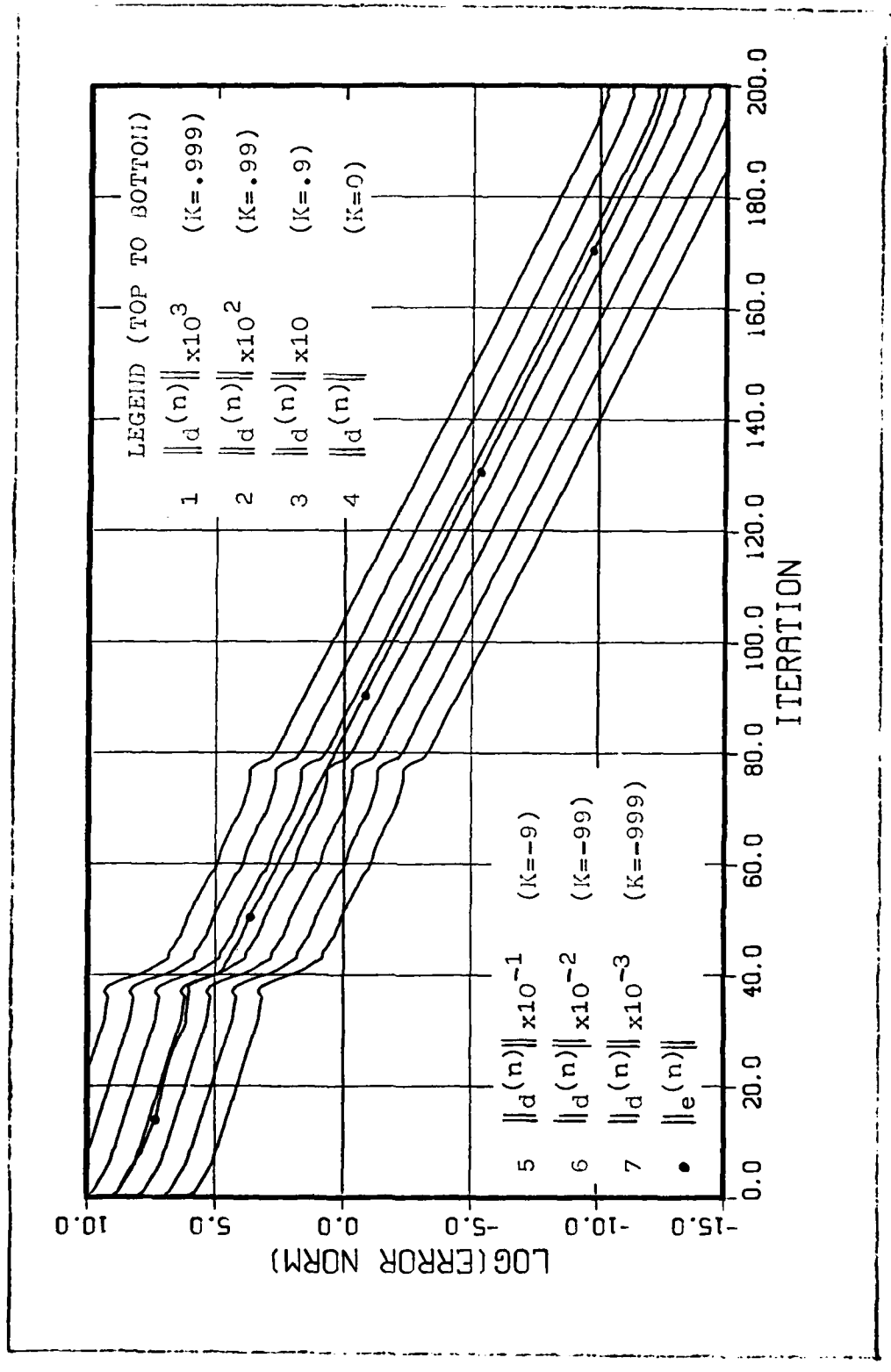


Figure 33. Order of Magnitude Comparison between Error Norm and Displacement Norm, Error Limit Method 1, 20 x 20 Nodal Density

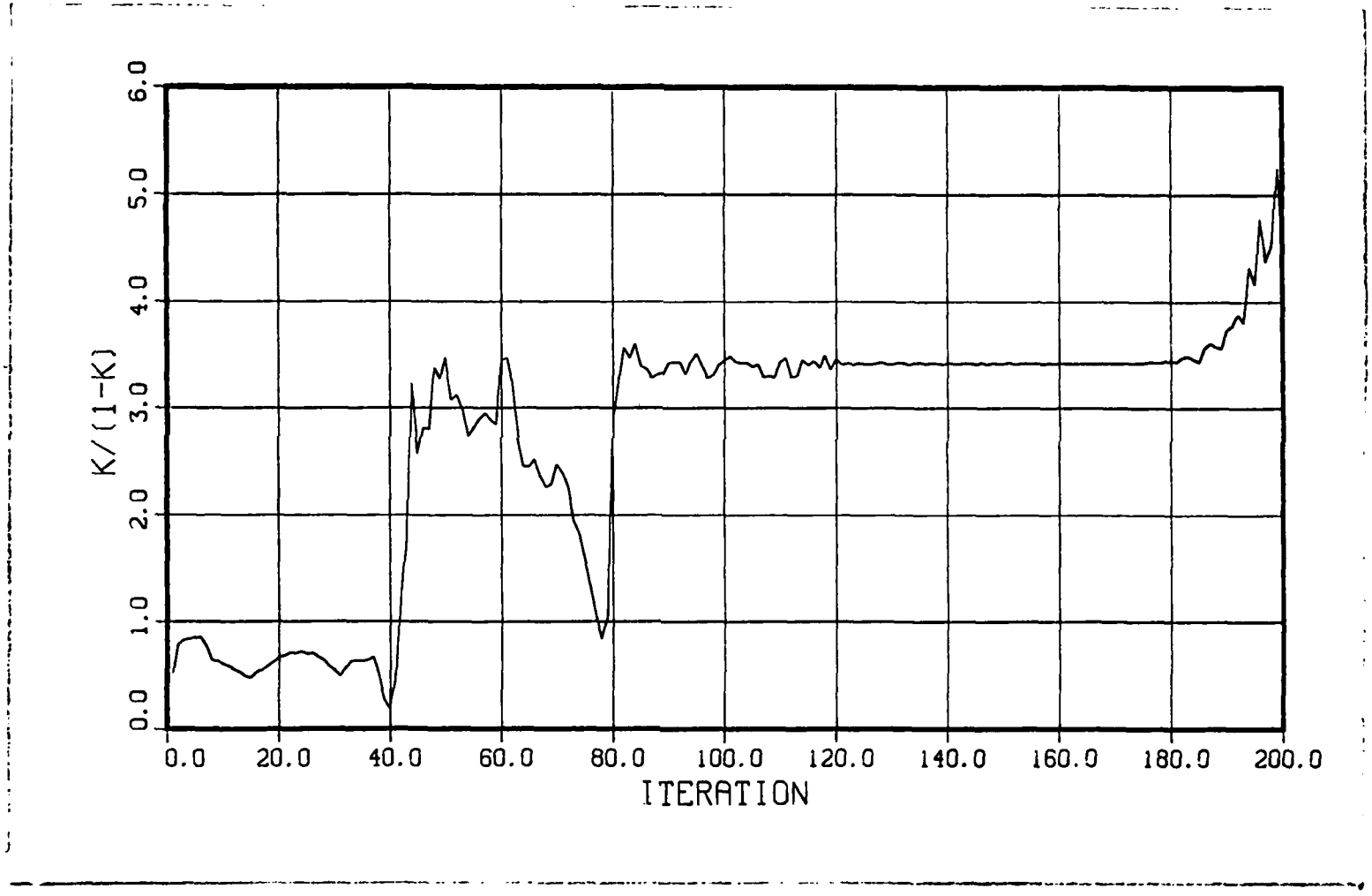


Figure 34. Error Norm to Displacement Norm Ratio, Error Limit Method 2, 20 x 20 Nodal Density

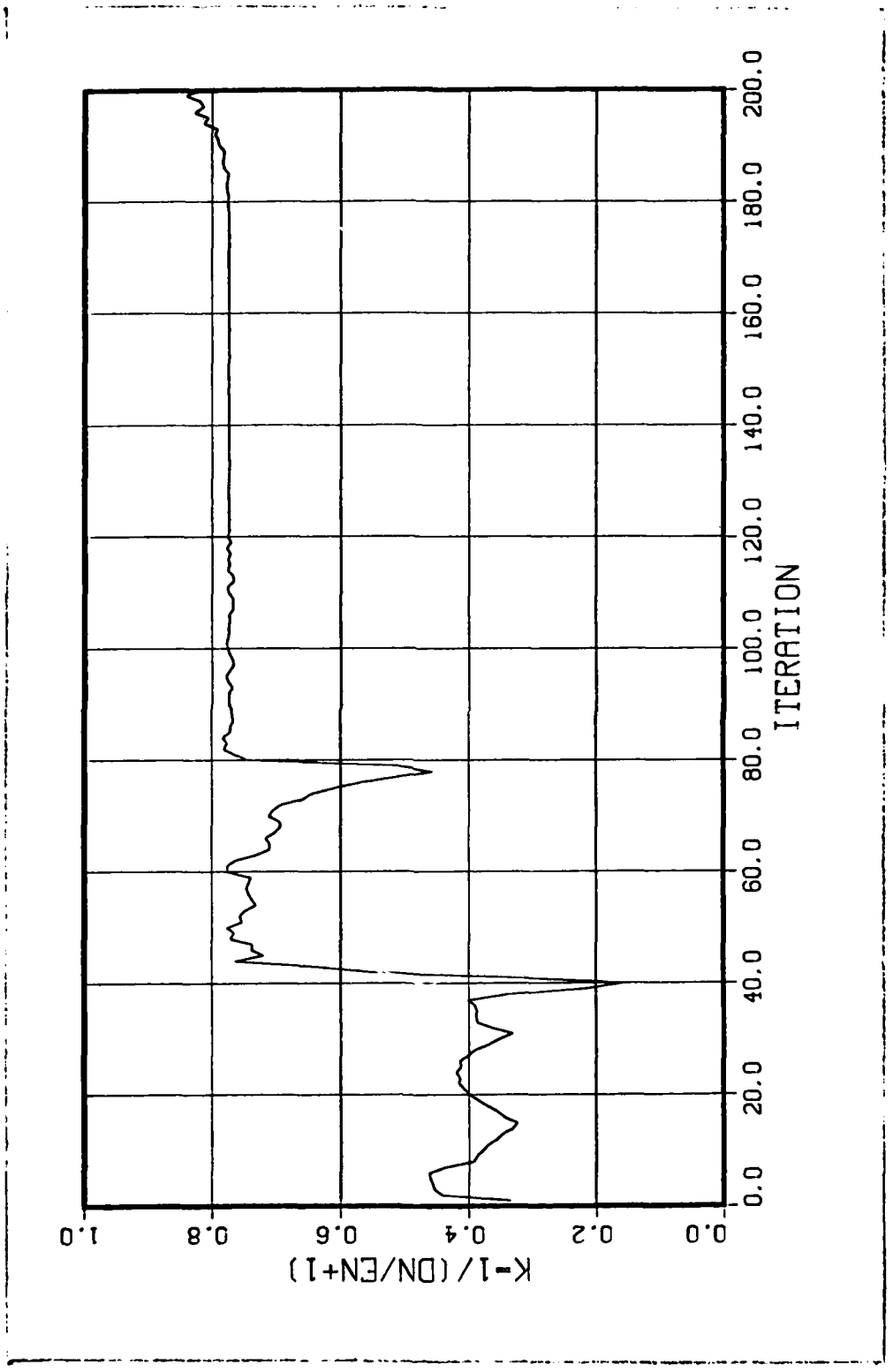


Figure 35. Parameter Required for Error Limit Method 2 to Equal the Error Norm, 20 x 20 Nodal Density

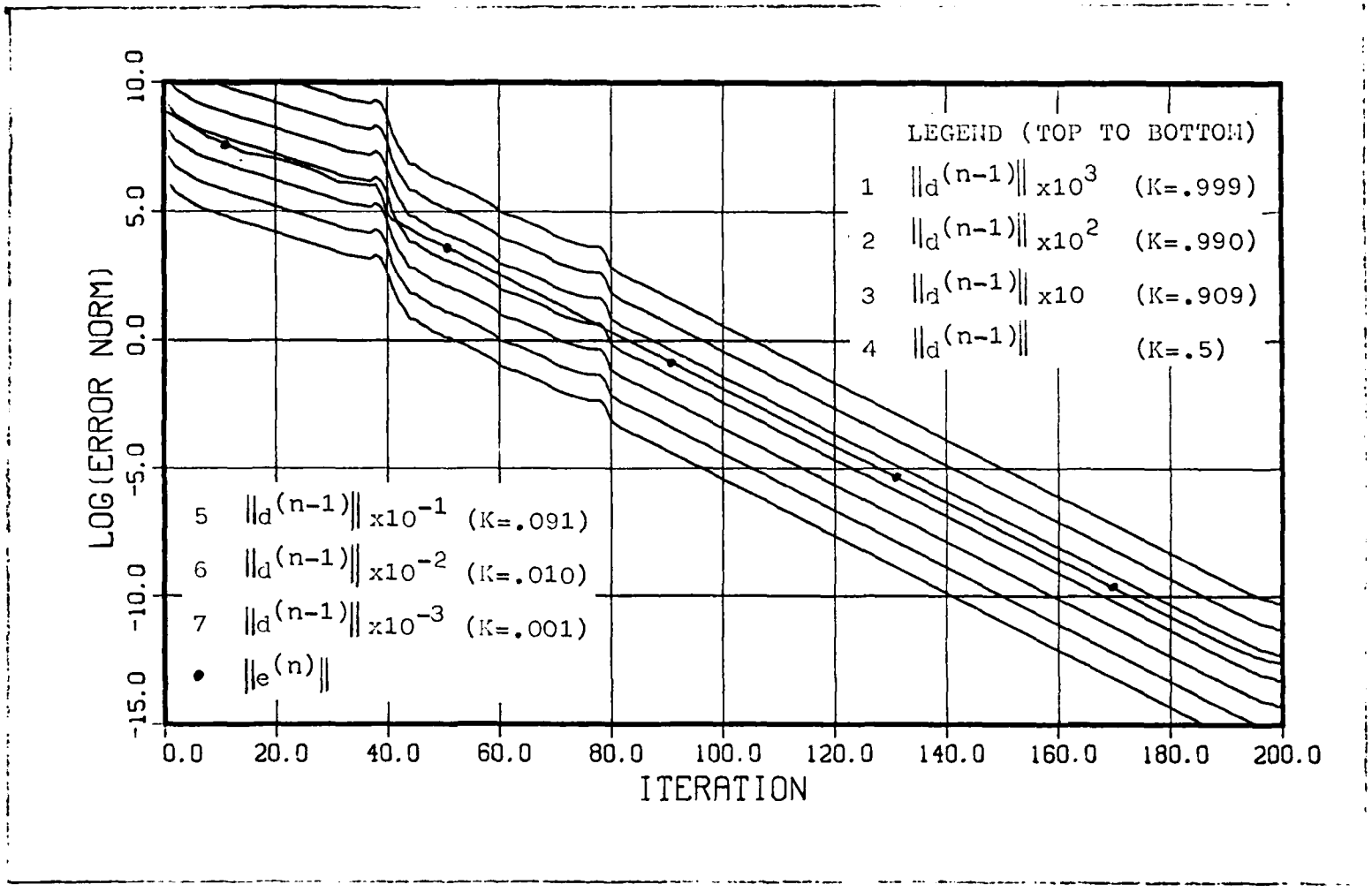


Figure 36. Order of Magnitude Comparison between Error Norm and Displacement Norm, Error Limit Method 2, 20 x 20 Nodal Density

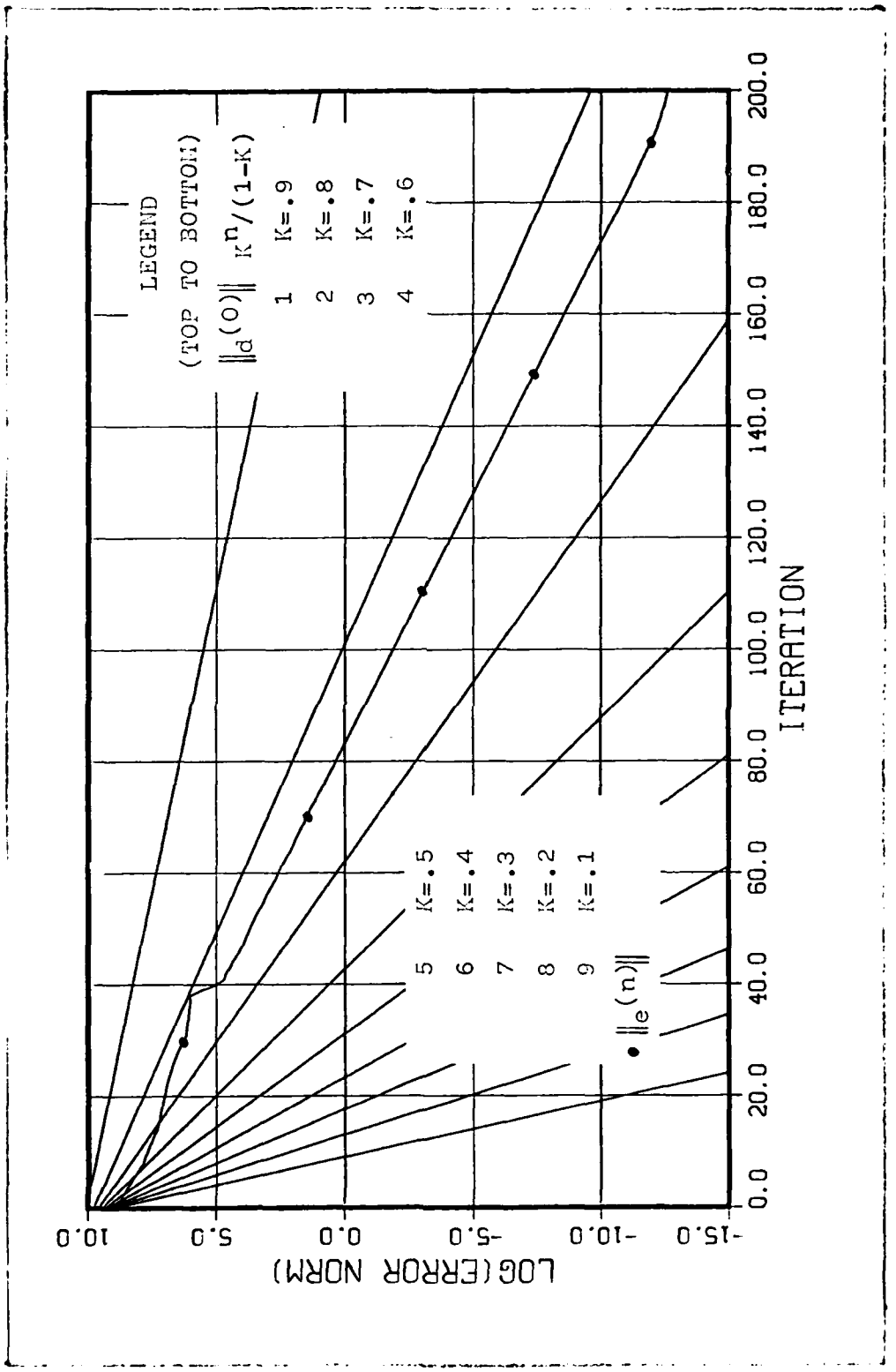


Figure 37. Comparison of Error Norm to Error Limit Method 3, 20 x 20 Modal Density

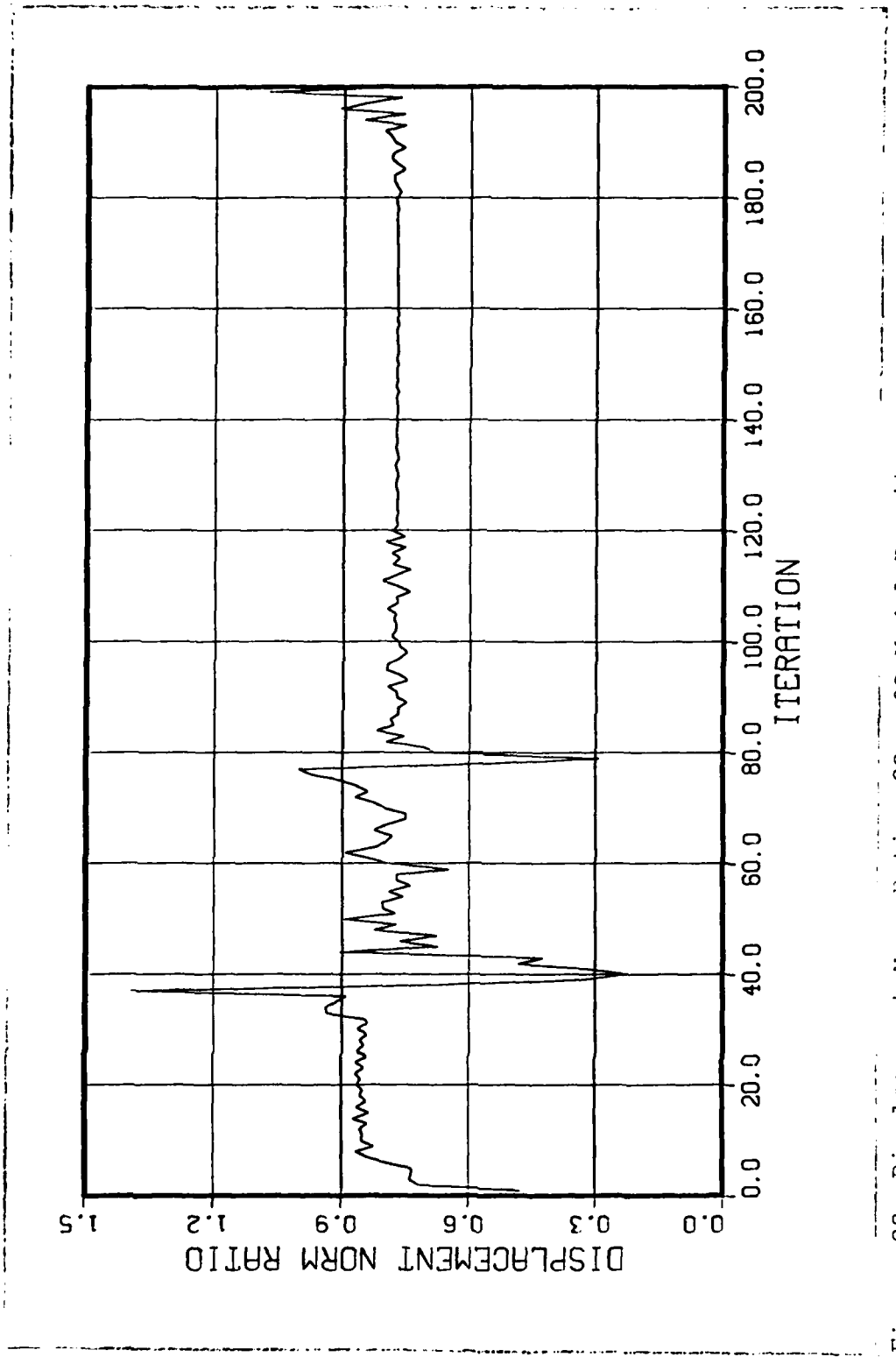


Figure 38. Displacement Norm Ratio, 20 x 20 Nodal Density

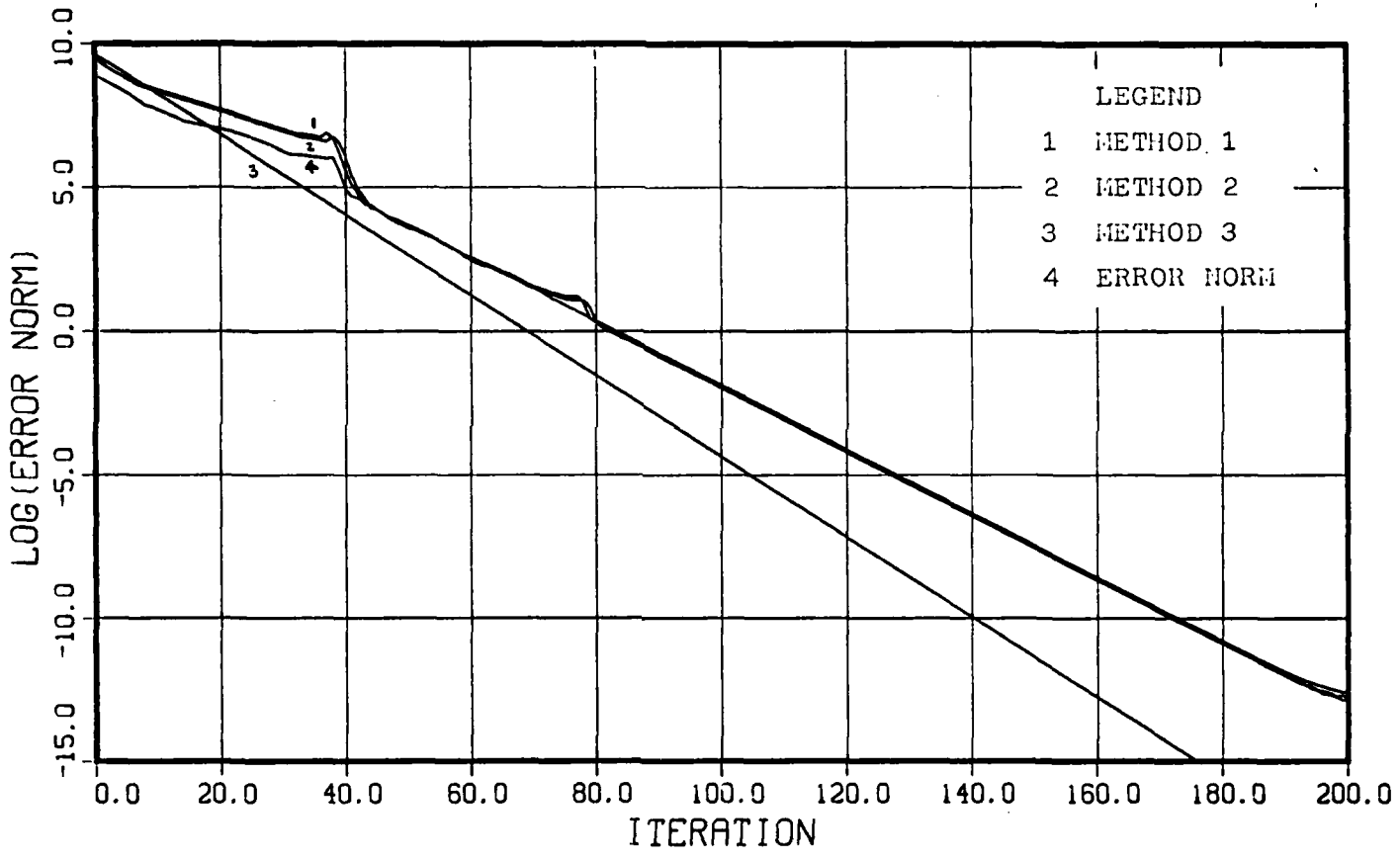


Figure 39. Three Error Limit Methods Compared to Error Norm, K = SOR Optimum Spectral Radius, 20 x 20 Nodal Density

Vita

Robert Alan Warren was born on 15 July 1949 in Somerville, New Jersey. He graduated from high school in High Bridge, New Jersey in 1967 and attended Stevens Institute of Technology from which he received the degree of Bachelor of Science majoring in Physics in May 1971. Upon graduation, he received a commission in the USAF through the ROTC program. He went on active duty September 1971 and completed navigator training receiving his wings in August 1973. He then served as a C-130 navigator in the 39th Tactical Airlift Squadron, Pope AFB, North Carolina until August 1977. He then went to Rhein Main AB, Germany and was a Standardization and Evaluation navigator in the 37th Tactical Airlift Squadron until entering the School of Engineering, Air Force Institute of Technology, in August 1980.

Permanent address: Fine Road
High Bridge, New Jersey 08829

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFIT/GNE/PH/82M-12	2. GOVT ACCESSION NO.: A115495
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): CONVERGENCE AND ERROR CRITERIA OF ITERATIVE NUMERICAL SOLUTIONS TO THE TRANSIENT HEAT CONDUCTION EQUATION	5. TYPE OF REPORT & PERIOD COVERED: MS Thesis
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Robert Alan Warren, Captain	8. CONTRACT OR GRANT NUMBER(s):
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Air Force Institute of Technology (AFIT-EN), Wright-Patterson AFB, Ohio 45433	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS: Materials Laboratory (MLSC), Wright Aeronautical Laboratory (AFWAL), Wright-Patterson AFB, Ohio 45433	12. REPORT DATE: March, 1982
	13. NUMBER OF PAGES: 167
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

15 APR 1982

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB, OH 45433

18. SUPPLEMENTARY NOTES: Approved for public release; IAW AFR 190-17

FREDRIC C. LYNCH, Major, USAF
Director of Public Affairs, AFIT

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Error Criteria
Numerical Convergence
Heat Conduction
Spectral Radius

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Full implicit finite differencing was used to approximate transient heat conduction in two spacial dimensions. The resulting set of linear simultaneous equations was solved using successive over-relaxation iterative methods. Errors between the exact matric solution and the iterated solution were computed and compared with several methods of determining error approximations that used the displacement between iterations and an associated parameter. Fundamental parameters of the iteration

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

matrix, such as the spectral radius, were tested as parameters in the error approximation methods. The error methods were compared with respect to accuracy, ease of computing, and computer resources required.

The nodal density used in the numerical approximation to the physical problem was compared with the spectral radius of the resulting iterative matrix problem. The spectral radius increases as nodal density increases.